High Quality Video Streaming with SCTP over CDMA2000

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Master Of Science
in the
University of Canterbury
by
C. Lee Begg

Examining Committee

Prof Krzysztof Pawlikowski	Supervisor
Prof Harsha Sirisena	Co-Supervisor
Prasan De Silva	Industry Mentor

University of Canterbury
2007

To Megan and my whole family.

Abstract

The research reported in this thesis investigates the performance of the transport layer Stream Control Transfer Protocol (SCTP) for streaming video over CDMA2000 cellphone and data wireless networks. The main measure of performance was quality of the received video at a given buffer size, as cellphones have memory of limited capacity. The hypothesis was that SCTP would be able to improve the quality of streamed video over UDP under the same memory requirements. Our study involved two series of simulation experiments and measurements in a testbed on the Telecom NZ CDMA2000 network, to test the performance of video streaming under SCTP and under UDP. It was found that SCTP did not improve the quality in streamed video with up to 5 second buffers. While other scenarios that have been tested by other people with high packet loss or congestion have shown that SCTP can improve the quality, the CDMA2000 network does not suffer from the impairments that SCTP could neutralise, and because of that, in this scenario, the quality of video streaming under SCTP and UDP are similar. The complexity that SCTP adds does not correspond to an increase in quality.

Table of Contents

Acknowledgments

This research project was partially supported by a Technology in Industry Fellowship (TNZX0501), in conjunction with FRST and Telecom New Zealand. The author wishes to thank his sponsors for making this thesis possible.

Chapter I

Introduction

Data streaming is the process of sending data across a network, during which the receiver can use the data as they arrive, without waiting for the entirety of the data to be received first. A memory space in a buffer at the receiver is used to smooth the flow of data to the rate required by the process consuming the data. Buffers introduce a delay in the start of consumption of the data, but allow variable inter-packet arrival times (called jitter) to not affect the consumer.

Streaming has become attractive solution for teletraffic consisting of video and audio data, so it is also known as multimedia streaming.

In general, possible ways of streaming data can be categorised by the reliability and if the stream is ordered. The reliability category differentiates between streams with full reliability, no reliability, and partial reliability. Reliability is gained through retransmission of missing data, which further delays transmissions. For partial reliability, a limited number of retransmissions are attempted, thereby limiting delaying the rest of the stream and not sending "out of date" data. The ordering category specifies if the data is provided to the upper layers in the same order it was sent in (ordered), if no order is enforced at all (unordered), or if the stream is partially ordered. Partially ordered means that some pieces are delivered in order and some can be marked by the application or upper layers to allow them to 'jump the queue'. One such use is when a high priority message is sent and needs to be actioned quickly: the packet can jump the queue, to become the next packet processed. By marking all pieces to 'jump the queue', the stream can become completely unordered. In summary, four possible categories of reliable/unreliable and ordered/unordered data delivery are possible:

1. Ordered/Reliable: as for example under TCP protocol[9] where all the data arrive in the order they were sent and the protocol keeps trying to re-send them if some data items fail to arrive,

2. Unordered/Unreliable: as for example under UDP protocol[21] where there is no notification or retransmission if delivery is unsuccessful and the protocol does not reorder the data so it can arrive in any order,

3. Partially Ordered/Reliable: where messages always arrive through retransmission but can be given to upper layers out of order if the packets were marked as such, and

4. Partially Ordered/Partially Reliable: where limited retransmissions are tried, and order is only enforced on the packets not marked.

Partially Ordered/Reliable transfer for multimedia teletraffic, such as images and audio, has been shown to be more efficient than the first two in particular network conditions[3, 4, 19]. Stream Control Transfer Protocol (SCTP) is a partial ordered/reliable transport protocol that also includes an extension for partial reliability; see section 2.3 for more discussion.

CDMA2000 wireless cellphone networks provide IP connectivity as well as voice services. The CDMA2000 1xEV-DO networks being deployed since 2002 have a maximum downlink data rate of 2.4 Mb/s, averaging at about 300–500 kb/s. The CDMA 1xRTT network offers average downlink data rate of 40–80 kb/s with a peak data rate of 150 kb/s. The underlying CDMA technology is likely to see much more use, and the CDMA2000 standards are going to be used for many years to come. As a current network access technology, one service of CDMA2000 is video streaming. See section 2.4 for more discussion about CDMA2000 and the CDMA technology.

1.1 Problem

Improving quality while minimising the size of the buffers for receiving data of multimedia streaming is important to achieve satisfactory performance and address scaling issues of online multimedia services. Some trade off of

quality is acceptable for some applications and investigations of effects related with partial reliability of delivery for multimedia streams are important for achieving better quality without trading too much buffering.

Mobile devices have limited bandwidth available to them, and have only a small amount of memory. Video streaming stretches both of these. The essence of the problem can be stated as trying to improve the quality of streamed video at a fixed data rate without increasing memory requirements or computational complexity too much.

1.2 Significance of Research

The main goal of this research project was to evaluate the usage of SCTP as a transport layer for streaming video over CDMA2000 1xRTT and 1xEV-DO wireless links. We considered its performance relative to UDP, and the memory usage and quality of each method. Both full and partial reliability cases with SCTP were tested against UDP streaming.

We wanted to identify which transport layer option offers higher quality for a given data rate and buffer size that the specific mobile terminal device type could cope with.

Our hypothesis was that SCTP with partial reliability could improve the quality of the video, but it could also increase the number of frames that need to be buffered for a short period of time. With full reliability the quality could be preserved but require a very large frame buffer. This led to two specific hypotheses is formulated in section 4.2.

1.3 Method

To investigate the behaviour of UDP and SCTP streaming on the CDMA2000 network, we performed a series of two simulation experiments as well as experiments using the Telecom New Zealand CDMA2000 network. The simulation experiments utilised the NS2 network simulation, with Akaroa2 controlling the simulation to produce low statistical error results. Several external tools from the EvalVid package were also used in both the simulation experiments and the experiment on the network. Chapter 4 explains how they were set up.

Results from the experiments are presented in chapter 5 and conclusions are given in chapter 6.

The research results can be immediately implemented in CDMA2000 networks, for use with PDAs and laptops that connect to the network and download video.

Chapter II

Background

The purpose of this chapter is to introduce the technological background of this thesis. The main issues discussed in this thesis are related with the H.264 video compression, the Stream Control Transport Protocol (SCTP) and the CDMA2000 wireless network. The stack of network protocols in mobile terminals considered in this thesis, from the bottom up is composed of CDMA2000, RLP, IP, SCTP and H.264.

2.1 Digital Video Basics

Digital video is a sequence of images shown in quick succession. The human visual system's perception of this is smooth continuous movement just like real vision, thanks to persistence of vision. The time difference between images can be as little at 0.1s for very slow movement; at time differences less that 0.05s (more than 20 frames per second) there is not an appreciable difference in quality. Most video systems use 15, 24, 25 or 30 frames per second. This rate for frames is called the frame rate.

2.1.1 Compression

Digital video has a lot of redundant information in it. These redundancy can be categorised into Intra-frame (inside a single frame) and Inter-frame (between successive frames). Different methods are used to encode these different redundancies efficiently.

Intraframe compression

When dealing with just one frame of a video, the compression doesn't differ from normal image compression. Compression techniques such as Wavelet and Discrete Cosine Transform (DCT) (like that found in JPEG image compression) are used. This is often referred to as texture encoding.

For lossy video compression lossy image compression is used for intraframe coding. Lossless video can use a number of different techniques to achieve accurate reconstruction.

Frames encoded when just using intraframe compression methods are larger than those that utilise interframe compression.

Interframe compression

Video data has a lot of temporial redundancy—each frame is closely related to the previous frame in most cases.

A small area of pixels in one frame will be closely related to another area in the next frame, therefore a prediction can be made when encoding as to the transformation between them. This concept is motion compensation, allowing for, and compressing better, the movement of the camera and objects in the frame. This is often expressed as Motion Vectors.

Motion Vectors are "pointers" to similar area to the current area in the previous, earlier or a later frame. The vectors are normally limited in how far they can point, to reduce the search space of the similar area and compression time, and keep the compressed size of the motion vector down. This form of motion compensation works well for objects moving inside the frame, a separate "global" motion conpensation method can be used to improve the compression over just motion vectors.

Areas that cannot be encoded by the motion compensation methods are encoded using intraframe compression, as are regions where the error in the reconstructed image is large.

Frame types

In most video compression systems, there are three types of frames; those that have just intraframe compression (I-Frames), those that are predicted

from a previous frame (P-frames), and those bi-predicted (bidirectionally predicted) from a previous frame and a future frame (B-frames).

I-frames are the largest, are the most accurate representation, and do not depend on previous frames for information.

P-frames are smaller but depend on the previous I or P-frame for some information. Made partly of motion vectors and the rest of texture encoding.

B-frames are the smallest, but depend on the previous I or P-frame and the next I or P-frame for information to predict the contents of the frame[23]. These frames are mostly motion vectors from either the earlier frame or later frame.

2.1.2 Frame Interleaving

Digital video is organised into a recurring sequence of frame types. These sequences are known as a Group of Pictures (GOP). Different standards and standards setting organisations call the Group of Pictures different things, such as Group of Frames (GOF), Group of Video Objects (GOV), and Group of Images (GOI)[8]. The concepts are similar but can have different features.

Sequence	Length	# of B Frames	Example/Usage
I	1	0	Studio video, Motion-jpeg, high bit rate and quality, simple.
IP	2	0	Studio video, better compression.
IBPB	4	1	Very high motion, fast changing video.
IBBPBB	6	2	DivX high motion codec, good for action scenes.
IBBBPBBB	8	3	
IBBPBBPBB	9	2	
IBBPBBPBBPBB	12	2	DivX low motion codec, good for typical video.
IPPPPPPPPPPPPP	15	0	Video conferencing.

Table 2.1: Some example GOP sequences (in natural order) and their usages.

Groups of Pictures always start with an I frame, and have only one in each sequence. P frames occur at fixed intervals in the GOP, except for

where the I frame is at the start of the sequence. B frames fill in the rest of the sequence. Some example sequences are shown in table 2.1. Each sequence has its advantages. For example, a long run of P frames is useful for video conferencing because there isn't much motion, the bit rate should be minimised, and the delay must be kept low. B frames can't be used because they would increase the delay having to wait for the next P frame to arrive (see the next two paragraphs). At 15 frames per second, a sequence length of 15 is used, meaning that the large I frame is only sent every second to correct any errors that have accumulated through the P frames.

There are many orderings of the frames within a sequence. The order the frames are shown is the order they are displayed in as they are played, which is referred to as the natural order of the frames in the sequence. However, in many standards, natural order is not the order the frames are encoded, decoded, stored, or transmitted in. Frames contain their frame number, timestamp or other information for returning them into natural order before they are played.

To encode (or decode) a GOP, the frames have to be encoded in the order from the least dependent frames to the most dependent, namely, I, P then B frames. For the GOP sequence of $IBBPBB$ (written with GOP numbers as $I_1B_1^0B_1^1P_1B_1^2B_1^3$ when the sequence number is 1), first the I frame is encoded. Next are the previous GOP's last two B frames, as they depend on the previous P (or I) frame and this I frame. Following those, is the P frame, then the first two B frames of this GOP. This gives an output of frames in the order of $I_1B_0^2B_0^3P_1B_1^0B_1^1$, referred to as the encoding order. The decoding occurs in the same order, as often does the storage or transmission.

Interleaving GOPs

When transmitting across networks using reliable or partially reliable transports, retransmission can occur if a packet is lost by congestion, error, or routing. These retransmissions occur about $2 \times RTT$ (Round Trip Time) later than the original transmission, or later if flow control or congestion control is involved.

If a frame of video is received after the time it was required to be decoded

before being displayed it is discarded. When a frame is discarded, the quality of the video displayed is decreased, and any frames that depended on this one also have reduced quality. A frame arriving before it is required to be decoded is buffered until needed. The number of frames in the buffer should be as small as possible to minimise memory usage, but should be large enough that the buffer is not filled completely—frames received when the buffer is full have to be discarded. This buffer can also be used to reorder the frames into the encoding order.

2.1.3 MPEG 4

MPEG 4 (Moving Picture Experts Group standard number 4) is a digital video standard created by ISO (International Standards Organisation) in 1998 and further extended in 2003. It is the successor to MPEG2 and MPEG1 and improves on the compression and quality[20].

The standard defined the bitstream, not how the video is encoded. This allows many implementations, and different approaches to encoding, to get the best quality for a given bit rate. Profiles are used to choose which subset of features are used to reduce complexity in encoding and/or decoding, or to reduce the costs.

Error Resilience and concealment

The MPEG4 standard specifies a collection of methods for dealing with errors. These were tested with Bit Error rates up to 10^{-3} and burst errors up to 20ms long[20].

Two filters are used to conceal errors in transmission and encoding. The Deblocking filter is used to remove visual disparity between neighbouring blocks in the picture, caused by the Motion Vectors not referencing matching areas and missing (error or not received) texture blocks. The Deringing filter counter-acts the visual "ringing" of sharp edges in the image by the Discrete Cosine Transform (DCT) used for Texture encoding.

Synchronisation symbols are inserted at fixed intervals in the stream and are used to recover from missing or corrupted pieces of the stream. Older standards, such as H.263 and MPEG2, only allowed resynchronisation at I

frame—since they contain data for every pixel independent of earlier frames—but some of the encoding used in MPEG4 allow prediction of the missing data, or locate the start of the next piece of data that is decodable, such as a slice (part of a frame) or the next frame.

MPEG2 added the concept of enhancement layers, which are used to improve the quality of the video by adding a bit more information to the video stream[14]. A complex video server could remove them as needed to reduce the bitrate to a client. MPEG4 takes this one step further by introducing fine-grained enhancement layers. These new enhancement layers can be truncated at any point in the layer and still be a valid layer. This gives greater flexibility to the video server or processing service to provide a range of datarates, not just one or two.

H.264

The H.264 standard has many names due to being a joint standard between ITU-T and ISO. These names include: H.264 from the ITU-T, MPEG4 Part 10 from ISO and the Advanced Video Codec (AVC) from the Joint Video Team (JVT) committee of both ITU-T and ISO.

The standard, created in 2003, is based on the MPEG4 standard with a few improvements and implemented as an MPEG4 profile. H.264 specifies a different motion prediction for motion vectors which produces better compression, and therefore better quality at the same bitrate. Generalised B frames were added, which allows B frames to reference any recent (within bounds) frame for predictions to be made from instead of just the previous (and next) P or I frame[8, 14].

The standard was tuned to give better quality for low bit rates. H.264 decoders are expected on 3G/3.5G phones, due to the good quality at low bitrates.

2.2 User Datagram Protocol

User Datagram Protocol (UDP) is one of the oldest Internet protocol standards[21]. UDP only adds application multiplexing (using port numbers) and checksumming for the data before it is sent over IP. As such, UDP only adds 8 bytes

of overhead to the data and is lightweight. Compared to TCP, UDP does not provide ordering of data, flow control or reliability (although the whole packet must be received without errors to be passed to higher layers), and provides data framed into datagrams, not as a continuous stream as TCP does.

2.3 Stream Control Transport Protocol

Stream Control Transport Protocol (SCTP) is the third general purpose Transport Layer Protocol standardised by the IETF (Internet Engineering Task Force, part of the Internet Society), after TCP and UDP[25]. It was originally designed as a transport for Signalling System Seven (SS7) for telephone signalling across Internet Protocol networks, but has found some favour in wider circles. There are currently very few applications using SCTP, but its features are of interest in many problem domains.

The specific features of SCTP include:

- Multi-homing with automatic failover, allowing a host connected to two or more networks to continue to communicate if the primary network path should fail;

- Multi-streaming without inter-stream head-of-the-buffer blocking, allowing multiple independent communications inside a single association (connection);

- Message based framing, meaning that a message will be passed in whole to the upper layers, even if it spans multiple IP packets, and if there is multiple messages in a single packet they will be passed separately, not a stream of bytes like TCP;

- In-order and Out-of-order message queueing, giving partial ordering as introduced in section 1;

- Full and partial reliability options[24], see section 2.3.1;

- TCP-friendly flow control, see section 2.3.2;

- SACK and fast retransmission, giving less overhead and better performance than TCP; and

- Heartbeat monitoring of links, to determine when a network path or host is not available.

Of particular interest to this research are reliability options. Flow control may also influence the results of streaming using SCTP.

These features and differences from TCP have made SCTP suitable for different types of networks, including satellite networks[6]. Because SCTP is connection-orientated like TCP, it is similarly unable to be used for multicast.

2.3.1 Partial Reliability

Every message (called data chunks) sent using SCTP has a "lifetime". If the lifetime of a message has expired before an attempt to transmit it occurs, it is discarded, and a notification is sent to the application to indicate that the discarding did occur. The default is to have an infinite lifetime, and therefore the data chunk is always, eventually, sent.

The partial reliability extension for SCTP as defined in RFC 3758 defines a "timed reliability" service[24]. This service, when a data chunk is marked as partially reliable, extends the normal lifetime behaviour by not sending or retransmitting a data chunk if it's lifetime has expired.

2.3.2 Flow Control

The flow control and congestion control algorithms in SCTP are similar to that in TCP with Selective Acknowledgements (SACK). The "additive increase, multiplicative decrease" mechanism is used, as in the slow start, congestion backoff and fast retransmit phases.

Selective Acknowledgements are where ranges of packets past the current cumulative acknowledge point are acknowledged as well as to the cumulative acknowledgement point. This saves retransmitting packets which have already been successfully received. Selective acknowledgements are optional for TCP, whereas in SCTP it is required.

TCP's flow control and congestion control algorithms prevent networks from becoming completely overwhelmed with traffic, and share limited network bandwidth fairly.

A number of small changes have been made to the flow and congestion control to accommodate other features of SCTP. One such change is that 'Transport Serial Numbers' (TSN) are used instead of bytes for acknowledgements, as SCTP is a message based protocol, not a byte stream. This also has the side affect of allowing much larger congestion window sizes and longer intervals between wrap around of the acknowledgement number. This benefits satelite links with long delays and very high speed networks, which could not be fully utilised by TCP.

Because the flow control is similar to TCP, it is a so called "TCP-friendly" flow control. An SCTP connection and a TCP connection will fairly share a link because SCTP is TCP-friendly, whereas UDP is not TCP-friendly and doesn't share links fairly.

2.4 CDMA2000

CDMA2000 wireless cellphone networks provide IP connectivity as well as voice services. CDMA2000 is the brand name for a set of standards in which Code Division Multiple Access is employed, governed by the 3GPP2 project.

2.4.1 Code Division Multiple Access

Code Division Multiplexing (CDM) is the general term for transmitting several signals, on the same frequency and at the same time, using codes to separate the different signals. Code Division Multiple Access (CDMA) is a subset of CDM used for allowing many users to access the network simultaneously.

Two forms of CDMA exist. Synchronous CDMA is simpler to understand as it is synchronous and the codes are truly orthogonal. However this requires strict control of timing which is extremely difficult to achieve on mobile stations. Asynchronous CDMA is asynchronous, allowing any station to transmit at any time, but the pseudo-noise sequences are not orthogonal at arbitrary starting points. The signal strength as represented by its power

must be equalised between senders as to prevent any pseudo-noise sequence "drowning out" other signals. This is performed by a fast closed-loop power control system with the base station controlling the power of the mobile terminals. Fast power control is not necessary for the transmit power of the base station as it can use Synchronous CDMA and transmit all outgoing signals simultaneously with equal power.

When the base station is using Synchronous CDMA, it can use the orthogonal codes which are more efficient in use of bandwidth, and therefore it can transmit higher data rates than mobiles can. A limitation for each form of CDMA restrict the data rates for the mobile-to-base station link. Using Asynchronous CDMA with almost uncorrelated, but not truly orthogonal, pseudo-random noise sequences have to use a lower data rate because of retransmissions due to temporary correlations of the pseudo-random noise sequences. With Synchronous CDMA for the mobile-to-base station link, very strict timing is required, and because there is a variance in the distance from the base station, and inaccuracy in the timing of all the mobile nodes, the data rate has to be reduced to keep the signals from the mobile stations synchronised.

2.4.2 1xRTT

1xRTT stands for "1 times Radio Transmission Technology", and utilises one pair of 1.25 MHz bandwidth radio channels, operating in two different directions under Frequency Division Duplexing (FDD).

The 1xRTT standard was developed by Qualcomm and standardised by 3GPP2 as IS-2000. RTT networks offer average data rate of 40–80 kb/s with a peak data rate of 150 kb/s.

2.4.3 1xEV-DO

CDMA 2000 1xEV-DO Evolution-Data Optimized—formerly Evolution-Data Only—is a further development from the CDMA 2000 1xRTT standard. It doesn't directly support voice and is not backwards compatible with 1xRTT, but features much higher data rates.

EV-DO was standardised in IS-856 in 1999 by 3GPP2.

EV-DO networks currently being deployed have a maximum base station-to-mobile data rate of 2.4 Mb/s, averaging at about 300–500 kb/s. The mobile-to-base station link is 150 kb/s, but Revision A of the standard, being deployed as of 2006, increases this to 1.8 Mb/s.

Because 1xEV-DO starts with "1x", it also uses one pair of 1.25 MHz bandwidth radio channels like 1xRTT. Most EV-DO mobile devices are capable of connecting to 1xRTT networks as well.

2.4.4 Radio Link Protocol

A variant of Radio Link Protocol (RLP) is used in CDMA2000 to provide a Link Layer that can recover packet loss on the wireless network[15]. RLP is primarily used to reduce the problems with timeout and wireless losses in TCP, and effectively reduces packet losses from about 10^{-2} to 10^{-4}. Because CDMA2000's mobile to base station link uses Synchronous CDMA, with GPS as the timing source, they have lower bitrates, and therefore to avoid unnecessary usage of the mobile to base station link, Negative Acknowledgments (NACK) are used to solicit selective retransmission instead of positive acknowledgments[28].

Negative Acknowledgments are sent when the receiver notes that it has not received a packet. This is done by tracking the sequence numbers and when one sequence number is skipped, a NACK is sent to have the packet with that sequence number sent again. If there is a large pause in sending packets, a missing packet just before the pause will not be noted until the next packet comes. At the end of a connection, this 'next packet' never comes. To avoid this problem, the last packet is sent three times when a long pause in transmission is noted, giving the receiver three more times to send NACKs for missing packets.

2.5 Summary

H.264, UDP, SCTP and CDMA2000 are the main protocols considered in this study. The properties of the H.264 video compression standard have advantages over other standards and it is set to be common on cellphones in the near future. SCTP is a new network transport protocol with features,

such as partial ordering and partial reliability, that make it attractive to transporting video streams. CDMA2000 is a wireless networking standard utilising Code Division Multiple Access which allows multiple use of wide area cellphone and mobile data terminal networks. In this thesis we are assuming that these protocols are used together in a stack to provide a video streaming service to mobile terminals with high video quality using either UDP or SCTP as the transport layer.

Chapter III

Research Issues

This chapter surveys and specifies the research issues, problems and general model for the research reported in this thesis.

3.1 Limitations in Previous Research

There has been a number of research papers published on the topic of multimedia data streaming over Partially Ordered transport protocols. This section highlights some research results and issues arising from them.

3.1.1 Image streaming Over SCTP

Streaming images over SCTP on high loss (battlefield) networks has been studied, e.g. in [3], and it was found to be efficient and suitable in that environment. The results came from using the Partial Order features of SCTP. The study did not look outside this one media type, but did demonstrate streaming of multiple images simultaneously was possible with acceptable quality.

3.1.2 Partial Order Connection Protocol

Much of the initial work on Partial Order transmission occurred at the Protocol Engineering Lab at the University of Delaware[22]. The Partial Order Connection (poc) protocol was developed to show the advantages of this concept as a transport layer. This protocol was used to show that partial order transport is more efficient than either Ordered/Reliable (TCP) or Unordered/Unreliable (UDP) transports[4]. Their experiments included simultaneous transmission of images and audio, but did not cover any other

media types.

The development of SCTP occurred in parallel with the work conducted by the Protocol Engineering Lab until late in their research, thus no experiments were done that would have involved SCTP.

3.1.3 Streaming MPEG4 Video Over SCTP in Congested Networks

Results show that SCTP is suitable for streaming MPEG 4 video under normal to high congestion conditions on the network[19]. The research showed that partial reliability did improve the visual quality of the video using partial reliability as described in section 2.3.1 for I frames, and no reliability for P and B frames. In this case, I frames could be retransmitted as many times as necessary before a timeout expired. The research did not look at physical layer errors leading to frames being dropped, and CDMA may have significant differences in this regard.

3.1.4 H.264 over SCTP

A paper on streaming H.264 video (a standard profile of MPEG 4 video) over SCTP[1] showed that partial reliability of just one retransmission for I frames and for I and P frames provided better quality video playout than UDP. The paper also showed that the H.264 video coding method was better able to adapt to the flow control of SCTP and after congestion on the network passed, was able to reduce the buffering to a little less than what was required for UDP for the same congestion. When frame losses or congestion was high, much more buffering was needed for SCTP. With partial reliability for I frames, there is a chance of some I frames not arriving at all, lowering the quality.

3.1.5 Link Between Round Trip Time, Buffer Size and Quality

The relationship between RTT (Round Trip Time) and both buffer size and quality in MPEG4 streaming over SCTP was considered in [2]. As RTT increases, buffer size should increase too, to accommodate increasing jitter and retransmission delay of the packets containing the video data. Up to a

certain value for RTT, quality of the video stream is not greatly affected, but beyond that point it decreases markedly.

This research also considered partial reliability for I frames with just a single retransmission. A higher reliability for I frames will affect the buffer size and improve quality, but to what extent is unknown yet.

3.1.6 CDMA Video Streaming and Reliability Options

None of the papers cited above considered performance of video streaming in real networks. Physical media—especially in wireless networks using CDMA technology—could greatly affect the results due to them being different from the assumptions made in the simulations reported in [1] and [19].

There are a large number of reliability options that can be explored but have not been researched yet. For example, methods that have not been investigated yet include selection of the order in which the frames are sent in to increase chance for retransmission to occur before the frame is needed.

3.1.7 Packet Scheduling for Jitter Control

Methods for controlling jitter of UDP streaming packets at intermediate points (such as routers and gateways) in the network have been suggested, with the goal of dropping packets, as early as reasonable, if they are "running late" or have arrived much too early. This tries to address the problem of jitter, thereby decreasing the buffer size. These methods preform better with increasing the Round Trip Time, and hop count[2].

The method of controlling jitter does not work well with video traffic, as dropping an important packet may cause a much larger drop in quality than another packet, whereas audio traffic, used in the experiments reported in [2], does not have packets that are more important that others and packet loss does not affect past drop packet. The benefit of making UDP streaming TCP-friendly is also achieved by SCTP without modifying routers.

Reducing the Round Trip Time is a sensible approach, but can only be achieved through adding some delay and buffering somewhere else in the (end-to-end) system. A proxy close to the end point would help improve quality through lowering the delay in retransmitting the data that has been

lost by the physical layer interference.

3.2 Research Issues investigated in this thesis

On the basis of the reported survey, one can conclude that the quality of video streaming over the CDMA2000 1xRTT and 1xEV-DO networks if using SCTP, has not been investigated yet. The main goal of this thesis is to fill this gap.

3.2.1 Model

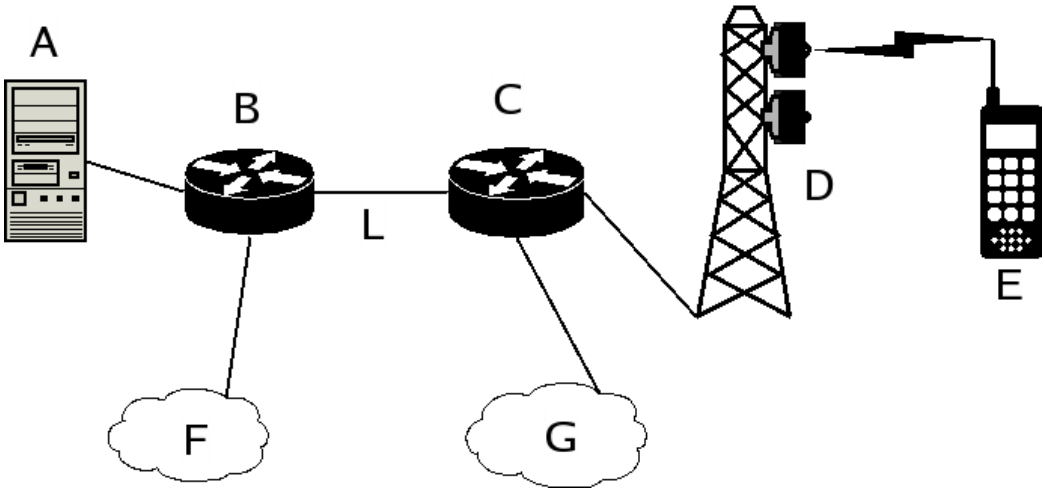


Figure 3.1: The assumed network topology. A, Video server; B and C, core routers; D, base station; E, mobile terminal; F and G, other user traffic sources and sinks; L, shared link.

We assumed a network topology depicted in figure 3.1, which should be of interest of typical cellphone network providers. The server providing the streaming of the videos—called a video server—is depicted on the left and labelled as 'A'. It is connected through two network routers, B and C, to the base station, D. The base station D transmits the data wirelessly to the mobile terminal E, using CDMA2000.

The video server A is centralised in the provider's network and is several routers away from the base station. The intermediate links are share with other services and users. The two clouds in figure 3.1, F and G, are used to

create a jittering base load on the shared link, L, between the two routers, representing the other services and users. Both directions are loaded with random traffic causing jitter.

We assume normal working conditions of the network, i.e. the links between the video server A and the base station D are not overloaded, however, jitter is observed.

The shared link L uses RED as the queue control method. All the wired links have delays of 10ms. The links are: 1 Mbit/s link to the video server and the base station, and 5 Mbit/s link for the base load nodes.

We also assume that there is no packet loss in the wired network, as such losses could be recovered by appropriate higher level techniques.

The four CDMA2000 wireless link bitrates were chosen to represent the average and best performances of 1xRTT and 1xEV-DO. This range of network bitrates was chosen to make projections about the bitrates between them, and are rates that are common in different scenarios. These bitrates are:

RTT-bad 76.8 kbit/s, the average 1xRTT throughput

RTT-good 134.4 kbit/s, the best 1xRTT throughput

EVDO-bad 480 kbit/s, the average 1xEV-DO throughput

EVDO-good 1248 kbit/s, the best 1xEV-DO throughput

The video data rates of 64 kbit/s, 128 kbit/s, 400 kbit/s and 1000 kbit/s were chosen to be common encoding rates with a lower bit rate than the responding wireless link bitrates.

3.2.2 Buffer Sizes

The buffer is used on the mobile terminal (labelled E in figure 3.1) to receive the data from the wireless network. Buffer sizes can vary significantly. We have considered seven cases with representative buffer sizes that could be used. This seems to be a sufficient number of cases to draw conclusions for the overall trends and interpolate for the buffer sizes not chosen.

Buffer sizes can be expressed in two ways:

1. by referring to their buffering capacity M_b, or

2. by referring to the time T_b needed to fill the buffer.

They are mutually related through the video's bitrate. In particular:

$$M_b = \frac{1}{8} \times V_b \times T_b \text{ [Bytes]}$$

and:

$$T_b = \frac{8 \times M_b}{V_b} \text{ [seconds]},$$

with M_b as the size of the buffer in Bytes, T_b as the size of the buffer in seconds, and V_b as the video's bitrate in bits per second.

Seven different buffer sizes in T_b were chosen in this thesis:

$T_b =$ **100 ms:** Buffer can store about 2–3 frames taking about 800 bytes of memory for 64 kbit/s video.

$T_b =$ **250 ms:** The initial playback is almost 'instant'.

$T_b =$ **500 ms:** The delay is very small and about typical of cellphones.

$T_b =$ **750 ms:** This is a possible buffer size for "smartphones", since they have more memory.

$T_b =$ **1 s (1000 ms):** This could be typical of PDAs, using 7.8 kByte to 122 kByte for 64 kbit/s to 1000 kbit/s video.

$T_b =$ **2 s (2000 ms):** A relatively long delay, typical of larger PDAs.

$T_b =$ **5 s (5000 ms):** Because computers have much more memory, this buffer size is common for video streaming of any kind and is about 610 kByte for 1000 kbit/s video.

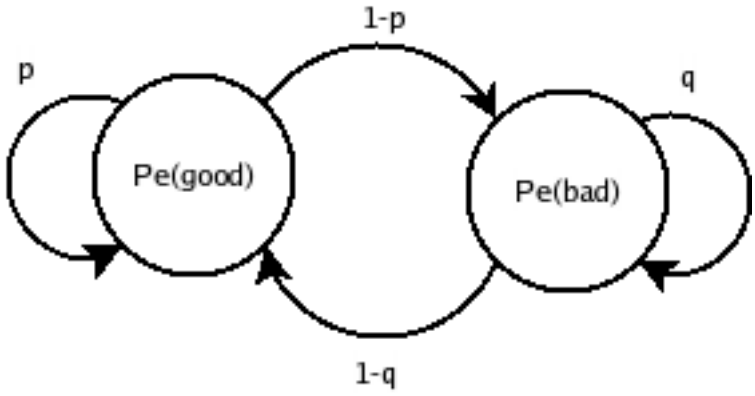


Figure 3.2: State diagram of a Markov chain used for modelling the packet losses in the simulation experiments

3.2.3 Modelling Packet Losses

The errors being modelled in our simulation experiments represent the packet losses over the CDMA2000 link between the base station and the mobile terminal.

The occurrence of errors in transmission over the wireless link was modelled by a two state Markov chain (implemented in ns2 as a "multistate error model" with two states) as shown in figure 3.2. It has two states: "good" (or "better") and "bad" (or "worse"). In the initial "good" state packet losses occur at a small rate, with probability $P_e^{(good)} = 0.01$. In the "bad" state packet losses occur with a high probability equal to $P_e^{(bad)} = 0.8$. This is a generalised case of the classical Markov model of errors introduced by [7] and used e.g. in [27, 17, 18], in which $P_e^{(good)} = 0$ and $P_e^{(bad)} = 1$. Our model of errors was considered e.g. in [29].

The probability of a transition from the "good" state to the "bad" state was assumed to be equal $1 - p = 0.02$, and the probability of a transition back to the "good" state from the "bad" state was equal $1 - q = 0.4$. These probabilities were assumed as representative after discussion with technical experts from Telecom New Zealand. A simple analysis of our Markov chain reveals that the probability of k consecutive packets being lost when the channel is in its "good" state is $(P_e^{(good)})^k = (0.01)^k$, while the same probability in its "bad" state is $(P_e^{(bad)})^k = (0.8)^k$. We can also determine the long

run probabilities of these two states (also called the steady-state probabilities), as $P(good) = 0.952381$ and $P(bad) = 0.047619$. One can argue that the rates at which the packets can be lost in our example are typical for links at licenced radio frequencies when they operate under CDMA.

3.2.4 Videos Samples

For this research, a set of videos was chosen so that they would be representative of what users would stream from an on-demand video service or user uploaded video service. Common videos would include news clips and similar videos like video blogs, and videos with large amounts of movement and complexity—such as sports and lifestyle videos. The range of videos was selected having in mind that it should have the main properties characterising different types of video scenes.

(a) CIF size frame (b) QCIF size frame

Figure 3.3: *Akyio* YUV video file frame 13 showing the relative size of (a) CIF and (b) QCIF size video at the same resolution.

The example videos used in this research were taken from the EvalVid corpus[26] and included the videos called *Akyio*, *Coastguard*, *News* and *Mobile*. All are 16 seconds long (400 frames) and had two sizes. The two sizes available were CIF (Common Interchange Format) at 352x288 and QCIF (Quarter CIF) at 176x144, both at 25 frames per second. The difference in size between CIF and QCIF when presented at the same resolution is shown in figure 3.3, and at the same displayed size with different resolutions in

3.7 (a) and (c). The two different sizes are both common, and sometimes it is preferable to have a smaller frame size scaled up for display because of the suppression of high frequency texture encoding and further distance in the scene that the motion vectors can cover. Both the reduction of high frequency textures and the greater portion of the scene the motion vectors can reach reduce the bitrate, as does starting with only one quarter of the data that the larger frame size has. On mobile phones, both sizes would be displayed full screen.

(a) Frame 1 (b) Frame 85

(c) Frame 117

Figure 3.4: Example frames from *News* CIF YUV video file.

Akyio and *News* are videos with television presenters presenting news. *Akyio* (shown in figure 3.3) has one presenter and a nearly static background, whereas *News* (shown in figure 3.4) has two presenters and a screen with fast action dance in the background (as can be seen in the subfigures a–c). These

videos are representative for news videos and similar programmes.

(a) Frame 1 (b) Frame 81

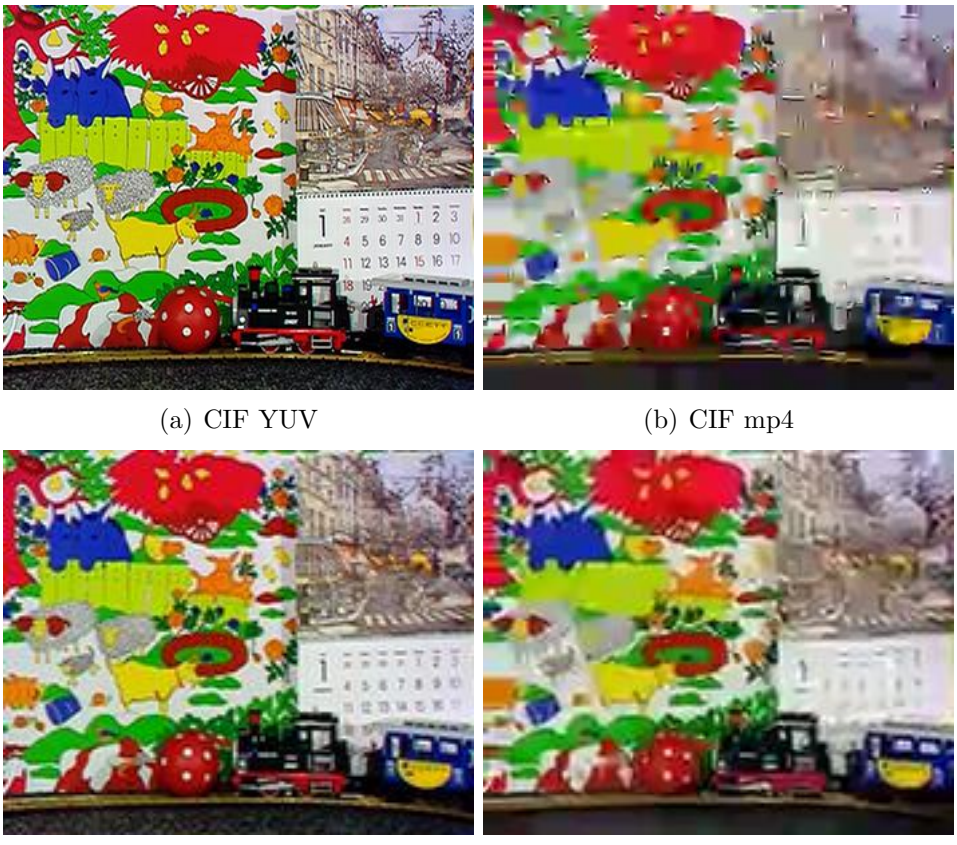
(c) Frame 161

Figure 3.5: Example frames from *Coastguard* CIF YUV video file.

In *Coastguard*, the camera pans following a boat in one direction, then a small coastguard boat in the other direction. This is similar to sport action and landscape vista captured by panning. Three frames from this video are shown in Figure 3.5.

Mobile is the most complex video in the EvalVid corpus. It has a collection of toys moving and a complex, but static, background, as shown in figure 3.6. The video has a much lower quality than other videos because of the high rate of movement and high frequency patterns, both requiring many bits in the bit stream to encode, and at low bitrates the video has very low quality, as the differences between the uncompressed (a and c) and the compressed video at 64 kb/s (b and d) in figure 3.7 shows. The smaller

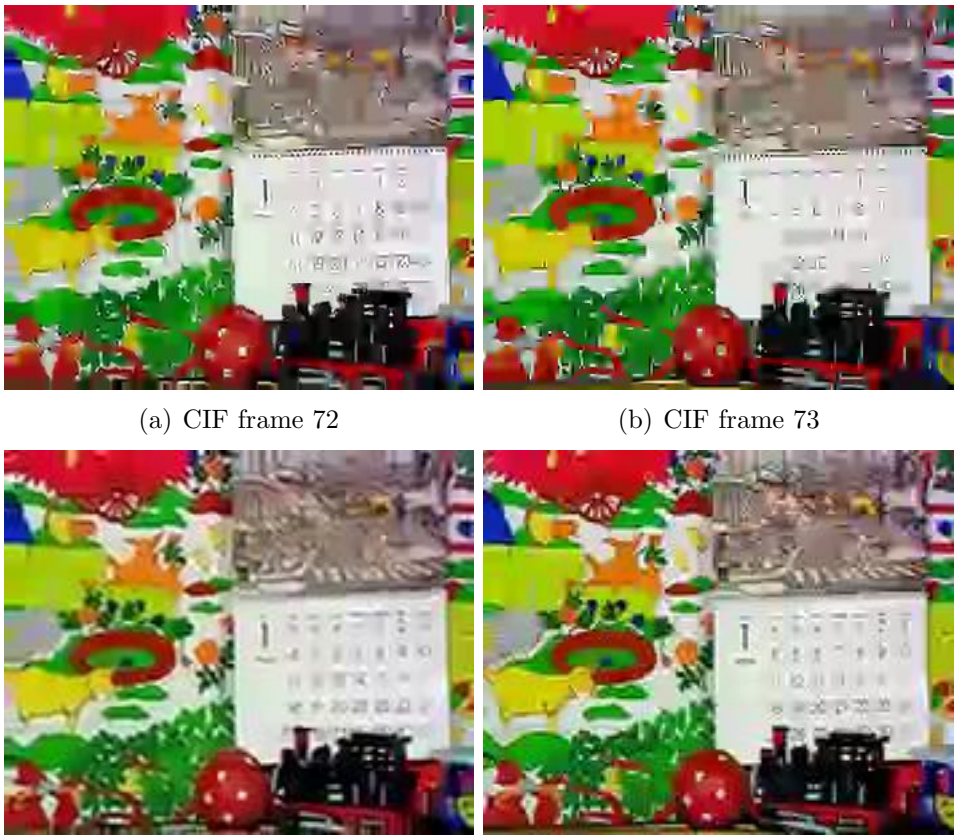
Figure 3.6: Example frame 200 from *Mobile* CIF YUV video file.



(a) CIF YUV (b) CIF mp4

(c) QCIF YUV (d) QCIF mp4

Figure 3.7: Frame 144 from *Mobile* CIF and QCIF YUV and mp4 at 64 kb/s video files.



(a) CIF frame 72 (b) CIF frame 73

(c) QCIF frame 72 (d) QCIF frame 73

Figure 3.8: Frames 72 (B frame) and 73 (I frame) from *Mobile* CIF and QCIF mp4 at 64 kb/s video files.

size QCIF video (d) appears better than the larger CIF video (b) for the compressed video, while for the uncompressed videos, the QCIF video (c) appears blurry compared to the CIF video (a). A particular visual quality problem exhibited by the compressed *mobile* video, especially CIF at 64 kb/s, is the "jump of blocking" in the image at each I frame, as shown in figure 3.8 with the previous frame on the left and the I frame on the right, for both CIF (top) and QCIF (bottom). The blocking of the frame of video is caused by the motion vectors selected imperfect blocks to take from an earlier frame. The jump of blocking is observed when an I frame is shown after another frame with bad blocking of the image, and the blocking caused by the 8x8 pixel texture encoding (see section 2.1.1) occurring in different places to the blocking caused by the motion vectors. Because it happens at every I frame, they occur at regular intervals with their lengths dependent on the GOP length: such as every 0.5 sec for GOP length 12 frames or every 2 sec for GOP length 50.

The complexity, high movement and high frequency textures present in the *mobile* video are common to many other contents of videos, such as mechanical systems, clockworks, sports events, crowds and some types of landscapes.

3.2.5 Measuring Quality

Measures of quality can be divided into two categories. The first category contains subjective measures, where individuals assess the quality on the basis of their personal experience. One of the most common subjective measure is created through a Mean Opinion Score (MOS) approach. The second category of measures of quality contains objective measures, where an algorithm is used to calculate a value of the measure. Note that the object measurements are repeatable—that is the measurement of quality can be repeated by someone else and the same results should be expected.

The basis of many objective measures is the Peak Signal to Noise Ratio (PSNR), itself a objective measure of quality. It measures the mean squared difference (MSD) between each frame in the measured video to the corresponding frame in the reference video on a per pixel basis, and is measured

in decibels. The mean squared difference for frame f of n pixels (MSD_f) is calculated as the sum of the differences between the kth pixel in the current (compressed or transmitted) frame (P_f^k) and the responding kth pixel in the reference (uncompressed original) frame ($P_f'^k$):

$$\text{MSD}_f = \frac{1}{n}\sum_{k=1}^{n}(P_f^k - P_f'^k)^2 \tag{3.1}$$

The PSNR for frame f is then calculated by

$$\text{PSNR}_f = 20\log_{10}\frac{\text{MAX}}{\sqrt{\text{MSD}_f}}\ [\text{dB}] \tag{3.2}$$

where MAX is the maximum value each pixel can take and is typically equal to 255. This is calculated for each frame. For the whole video, the individual PSNR_f values can be averaged to produce the average PSNR for the video, i.e.:

$$\overline{\text{PSNR}} = \frac{1}{F}\sum_{f=1}^{F}\text{PSNR}_f \tag{3.3}$$

where F is the number of frames in the given video.

While PSNR_f and $\overline{\text{PSNR}}$ do not accurately reflect the quality perceived by the human visual system, they are related with the perceived quality, and easily calculated—most video programs output the $\overline{\text{PSNR}}$ as part of the statistics for the compression.

The Mean Opinion Score (MOS) as mentioned is a subjective measure quality, that can be applied to video. The MOS was first used to describe the audio quality of telephone calls. The classic scenario for measuring MOS is to have a group of people which are shown or played the same video or sound, and asked to score the quality (or inversely, the impairment) of the voice or video between 1 and 5. The score equal to 1 means that the quality is totally unacceptable or the voice is impossible to hear, and a score of 5 means the quality is excellent, giving "crystal clear" sound or video. Because of MOS assumes a single number between 1 and 5, it is simple to understand, process and graph.

Because of the slowness and subjectivity of asking people to evaluate each

video, objective approaches to calculating the MOS have been sought after. One particular difficulty in creating an objective video quality measure is how to model or match the human visual system (i.e., the way the human eye sees things, and how the brain interprets the signals from the eyes), which is taken into account when the subjective MOS measure is performed. This has led to several different methods of calculating an objective MOS value and other objective measures, such as the `mos` tool from Evalvid, VQM, DVQ and VSSIM[16].

PSNR_f [dB]	MOS_f
> 37	5 (Excellent)
31–37	4 (Good)
35–31	3 (Fair)
20–25	2 (Poor)
< 20	1 (Bad)

Table 3.1: PSNR_f to MOS_f scale conversion. Modified from [10] (Table 2).

In the reported research experiments we used the method implemented in Evalvid 2.1, where values obtained for PSNR_f are translated into the 1 to 5 scale of MOS by applying Table 3.1 to give MOS_f, and then averaged over the whole video sequence[10], i.e.:

$$\overline{\text{MOS}} = \frac{1}{F} \sum_{f=1}^{F} \text{MOS}_f \tag{3.4}$$

In Evalvid 2.1, such modification of a $\overline{\text{PSNR}}$ is referred to as "`mos`". This measure was used in the reported research experiments.

3.3 Summary

Research has been carried out into many areas of multimedia streaming. The open issue that this thesis addresses is the quality of video streaming with the interaction between SCTP and CDMA2000 wireless networks. The assumed network scenario reflects a common situation for a CDMA2000 network provider. Having described the main research problem addressed by

this thesis, the network and video scenarios and research methodology, the next chapter will present the method used to perform the research.

Chapter IV

Experimental Design

This chapter presents the hypotheses investigated in the reported research project, our assumptions and the method of the experiments which were performed.

4.1 Aims

The aim our of experiments was to show whether SCTP or UDP would be better for streaming video to mobile terminals over the CDMA2000 networks. To do this, three experiments were devised.

Experiment 1 was to see if the encoding pattern—the GOP length, and the number of B frames—made a difference to the quality of video streamed. This experiment was done by conducting simulations based on ns2 which is a open source network simulator. It was also used to check the simulated network topology and models for errors, and to reduce the number of video-bitrate-network combinations that needed to be tested in the next experiments. The three encoding patterns chosen were common patterns. They were GOP 30 B 0, GOP 12 B 3 and GOP 12 B 2.

Experiment 2 was also conducted by simulation; this time to compare UDP streaming against SCTP streaming. Three configurations of SCTP streaming were tested: (i) full reliability, (ii) partial reliability of B frames, and (iii) partial reliability for P frames and B frames. The simulated network topology was slightly different to that used in Experiment 1, as described in section 4.3.6 due to the noted results from Experiment 1, see section 5.2.1.

Experiment 3 used a testbed on a real CDMA2000 1x EVDO network to validate the results of the second experiment.

4.2 Hypotheses

The experiments were designed to prove, or disprove the following two hypotheses:

Hypothesis 1:

> There is no statistical difference in quality between the encodings of GOP 30 B 0, GOP 12 B 3, and GOP 12 B 2 when transmitted over UDP on RTT and EVDO networks without congestion.

This hypothesis was tested by a series of simulations as outlined in section 4.3 in our Experiment 1.

Hypothesis 2:

> SCTP transport with partial reliability for B frames will perform statistically better than SCTP with partially reliable P and B frames, and UDP, for all buffer sizes. SCTP with full reliability will preform the worst, except for the buffer sizes over 2000 ms.

This hypothesis was tested in a series of simulations as outlined in section 4.3 in our Experiment 2.

These simulations covered a wide range of parameters. The simulation results were later validated in an additional experiment, Experiment 3, conducted in a CDMA2000 network testbed. In the testbed only UDP and SCTP with full reliability transport layers were tested, and only one network capacity was considered; see section 4.4 for more.

4.3 Simulation Experiments

The simulation .tcl files for `ns2` were set up for each video. `Akaroa2`[5] was used to launch the simulations and collate the results with appropriately low statistical errors of the estimate of $\overline{\text{MOS}}$ with less than 1% of relative statistical error (at 0.99 confidence level). The figure 4.1 shows the flow of data and the processes from source files and parameters to the results.

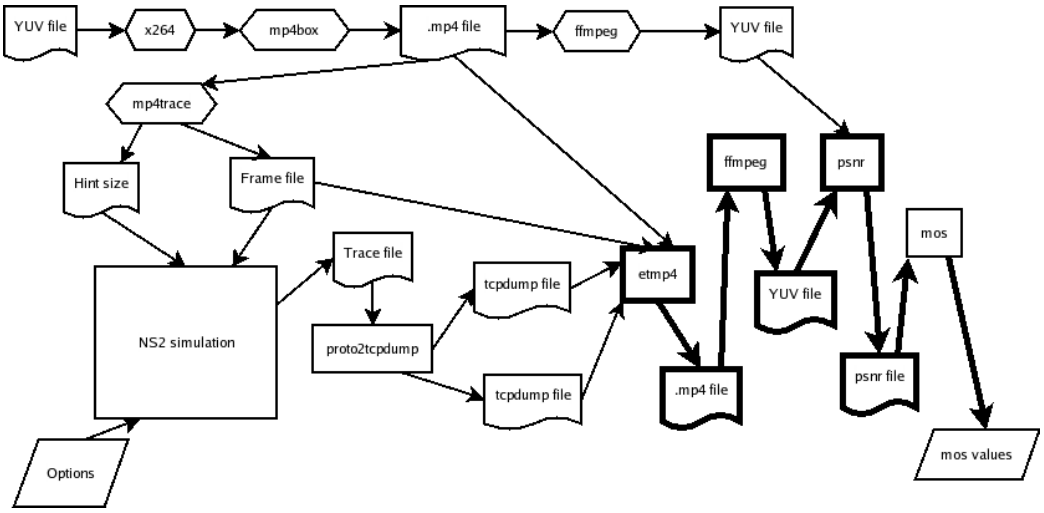


Figure 4.1: The flow of information through the simulation. Bold outlines and boxes indicate multiple instances. The terms are explained in the text.

4.3.1 Setup

The source video described in section 3.2.4 were provided in the lossless YUV format.

To set up each video in .mp4 format, the original YUV file was encoded (in two pass encoding) using `x264` to produce an H.264 elementary stream. `mp4box` was used to add a "Hint" track and pack it into .mp4 file (MPEG 4 part 14 video file container). The .mp4 file was used by `mp4trace` and `ffmpeg`. The first run of `mp4trace` produced a frame file, which contains the frame type, size and time. The size of the header/hint packet was extracted from a second run of `mp4trace`, this time in packet mode. `ffmpeg` was used to output an YUV file for the encoded file, identical in content to the .mp4 file but uncompressed. This file was compared to the original YUV file using `psnr` to produce a reference psnr file.

In summary, the inputs into each simulation are constructed from:

- tcl file, for seating up and running a given simulation;
- the original YUV video file, set in the tcl file as a filename;
- the .mp4 file, set in the tcl file as a filename;

- the reference psnr file, set in the tcl file as a filename;
- the frame file, set in the tcl file as a filename;
- the hint packet size, set in the tcl file;
- the type of network to use, set in the tcl file;
- the transport layer to use and options, set in the tcl file.

For each set of options and videos, a tcl file was created with the parameters to be used set, to instruct `ns2` to run the simulation and to perform the further processing, as well as to pass the current MOS values to Akaroa.

4.3.2 Simulation Run

In each simulation run the following steps were taken:

1. The simulation was set up in `ns2` and the topology created.
2. The simulation was run, producing a trace file.
3. The trace file was parsed by `udp2tcpdump` or `sctp2tcpdump` programs (labelled together as `proto2tcpdump` in figure 4.1) to produce two tcpdump files. Other parameters are also passed to the programs.
4. The following substeps were run once for each buffer size in each run.
   (a) `etmp4` was run taking the frame file, mp4 file, the two tcp dump files, and the buffer size, creating a 'traced' mp4 file that represents the data that was received.
   (b) The new traced mp4 file was decoded to a YUV file by `ffmpeg`.
   (c) The YUV file was compared to the original YUV file using `psnr` to create a psnr file for this buffer size.
5. The collection of psnr files was compared to the reference psnr file (that was created in the setup stage) using the `mos` program to produce $\overline{\text{MOS}}$ for each buffer size.

The $\overline{\text{MOS}}$ values were then passed to `Akaroa2` for analysis.

The programs `mp4trace`, `etmp4`, `psnr` and `mos` were taken from EvalVid 2.1. The `udp2tcpdump` and `sctp2tcpdump` were written specially for this research and are given in Appendix B.1. These two programs take the `ns2` trace file, and produced the sender and receiver tcpdump files that `etmp4` requires to process the 'traced' mp4 video file for evaluation. There was no program available that could convert from `ns2` trace file to tcpdump for both UDP and SCTP.

The simulation runs were controlled by tcl scripts run in `ns2`.

4.3.3 Control of Simulation Runs and Analysis of Simulation Output Data

`Akaroa2` is an automatic launcher of multiple replications in parallel, and on-line controller of simulations runs[5]. It starts multiple simulation runs simultaneously across a number of computers and keeps them running until the errors of results have fallen under a given level. For the simulation experiments the following parameters were specified:

- Precision of the final results (the maximum relative error allowed) = 0.01 (1%)
- Confidence level of the final results = 0.99 (99%)
- Minimum number of replications of simulations required = 100

These mean that our final results might have an error of no more than 1% at 0.99 confidence level.

The minimum number of replications was set to not miss any non-typical behaviour such as an uncommonly long run of dropped packets on the wireless network.

4.3.4 Simulated network topology

The topology in section 3.2.1 was used in the simulation with minor changes.

The two clouds in figure 3.1 were used to create a jittering base load on the shared link between the two routers. The base load was generated by

`Application/Traffic/CBR` in ns2, with a packet size of 500 bytes and an interpacket interval of 5 ms. Randomisation was turned on, so the interpacket interval were random, with a mean duration of the time interval between transmissions of two consecutive packets equal 5 ms. This was equivalent to a data rate of 800 kb/s (100,000 Bytes per second). Both directions—from B to C and C to B in figure 3.1—are loaded with this traffic.

The shared link uses RED as the queue method. All the wired links have delays of 10ms. The links are 1 Mb/s to the video server and the basestation, and 5 Mb/s for the base load nodes.

An extra link was added between the RLP wireless link and the video receiver with very high bandwidth and very low delay, because the arrival time and id of the UDP and SCTP packets were not getting writting to the ns2 trace file. This new link forced ns2 to write them correctly to the trace file. This could be assumed to be a processing delay within the mobile terminal passing the data between the CDMA2000 receiver and the video decoding unit.

4.3.5 Experiment 1

The two representative videos selected for this experiment were *akiyo* and *coastguard* (see section 3.2.4). They were encoded in two passes with the different GOP length, number of B frames, and bit rates using `x264` program with the default settings. The simulation only tested UDP streaming.

The method for this experiment was the simulation method shown in section 4.3. The shared link, L in figure 3.1, in the implementation of the model was fixed at 1 Mb/s for all simulations for this experiment.

4.3.6 Experiment 2

The four videos were selected for being representative, being the best static quality (*akiyo*), worst static quality (*mobile*), and two median static quality videos (*coastguard* and *news*); see section 3.2.4 for a description of each video. They were encoded in two passes with the different bit rates using `x264` program with the default settings and GOP 12 and 3 B frames. UDP streaming was tested, along with full reliability SCTP, B frames partially

reliable SCTP and, B and P frames partially reliable SCTP.

We simulated the network depicted in Figure 3.1. To prevent the dropping of packets in the wired network, the data rate of the shared link L was set to the rate of the video plus 1 Mb/s for each simulation run in this experiment.

4.4 Experiment 3: Experimental Testbed

The videos used in Experiment 2 were taken to be streamed from a video server to a laptop connected to CDMA2000. An altered version of `mp4trace` was created to stream videos using SCTP. For simplicity, full quality SCTP was used and it only required changes to one function; given in Appendix B.3. The altered version of `mp4trace` was used to stream SCTP, and the unaltered version of `mp4trace` for UDP.

`tcpdump` was used at both the video server and the mobile terminal to collect the timing and packet size information from the streaming sessions. The SCTP tcpdump files were processed and transformed into a format that `etmp4` could understand, using the `sctp2udpdump` program written for this research and give in Appendix B.2. The tcpdump files then used as input into the same process, written in tcl, that processed the results for the simulation experiments, starting at step 4. The results from 4 runs from different times of day and weekend—weekday morning, weekday afternoon, weekday evening and weekend—were averaged and standard deviations were calculated.

The video server was located in the Computer Science and Software Engineering Department of the University of Canterbury, connected to the university network at 10 Mb/s. The mobile terminal receiving the video was a laptop running Linux and had a Sierra Wireless AirCard 580 plugged in to give access to the Telecom New Zealand CDMA2000 1xEV-DO network.

4.5 Experimental Environment

This section highlights the assumptions made in the experiments performed. Assumptions relating to the model used are in section 3.2.1, and the assumptions related to the video used are discussed in section 3.2.4.

4.5.1 Simulation

In our simulations experiments, we used:

- ns2 version 2.29 with patches for Akaroa2 and RLP applied;
- the ns2 model for RLP, considering it as a representative of that used in CDMA2000;
- the error model for the CDMA2000 link, as described in section 3.2.3, considering it as a good representation of the real error pattern in CDMA2000 networks.

4.5.2 Experimental Testbed

Because this experiment was conducted in a live CDMA2000 network, we assumed that:

- the networks between the video server (at the University of Canterbury) and the CDMA2000 network (Telecom) are not congested and only induce jitter;
- the Round Trip Time between the video server and the mobile terminal is not significantly higher in this experiment than the round trip time would be for a local video server.

Chapter V

Results

5.1 Results

All detailed results from our two simulation experiments were obtained with the relative error smaller than 1% at a confidence level of 0.99. The testbed results are shown with their standard deviation.

Numerical results for Experiment 1, 2 and 3 are shown in Appendix A.1, A.2 and A.3, respectively.

5.1.1 Experiment 1

The results show the same general trend, up to a limit, over the increasing buffer sizes. All the graphs given here show typical results.

A graph comparing the two videos, *akiyo* and *coastguard*, at 64 kb/s on the RTT-bad network with GOP 30 B 0 encoding (GOP length 30 and 0 B frames), is given in figure 5.1, and shows a general rule that the *akiyo* video, with its lower motion and near perfect quality, has a much better $\overline{\text{MOS}}$ value than *coastguard*, for all buffer sizes.

Figure 5.2 shows the quality of the different encodings for *akiyo* at 64 kb/s at different buffer sizes. The $\overline{\text{MOS}}$ value for GOP 30 B 0 quickly rises from about 4.2 at the buffer size of 100 ms, to just under 5.0 for from 1000 ms onwards, whereas GOP 12 B 3 and GOP 12 B 2 start around 3.65 and approach their maximum about 4.0 from approximately 750 ms onwards.

Shown in figure 5.3 is the difference between the different bitrates for *akiyo* on the evdo bad network. All the videos with a bitrate of 128 kb/s and the 64 kb/s GOP 30 B 0 encoding all reach the maximum 5.0 $\overline{\text{MOS}}$ score. The next group of values down are the GOP 12 B 3 and GOP 12 B 2 results for 64 kb/s video bitrate. The low three results are for the videos encoded

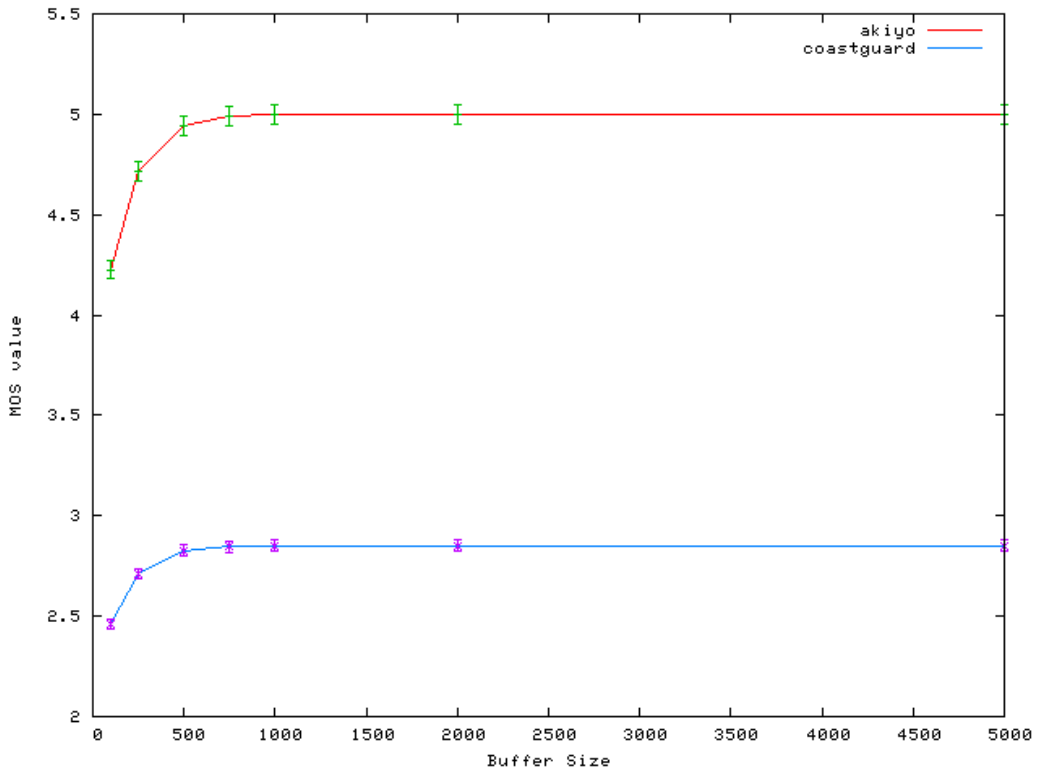


Figure 5.1: $\overline{\text{MOS}}$ for *akiyo* and *coastguard* at 64 kb/s on the RTT-bad network with GOP length and 0 B frames.

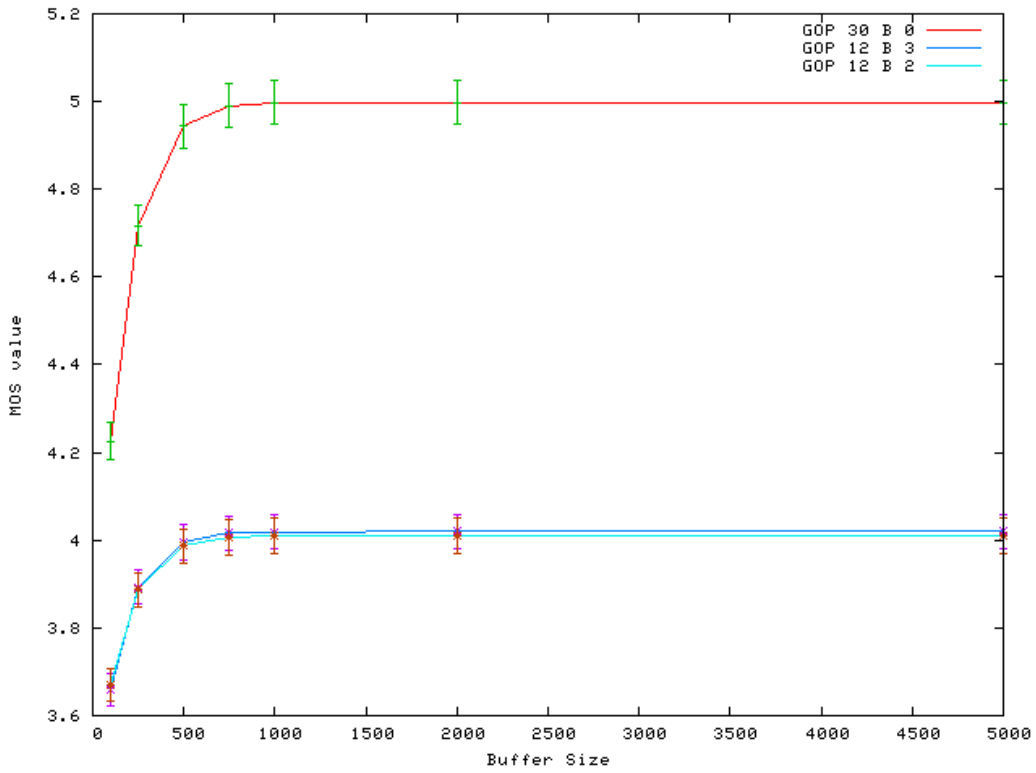


Figure 5.2: $\overline{\text{MOS}}$ for different GOP length and number of B-Frames encodings in *akiyo* at 64 kb/s on rtt-bad network. Encoding are GOP length 30 with no B frames (GOP 30 B 0), GOP length 12 with 3 B frames (GOP 12 B 3) and GOP length 12 with 2 B frames (GOP 12 B 2).

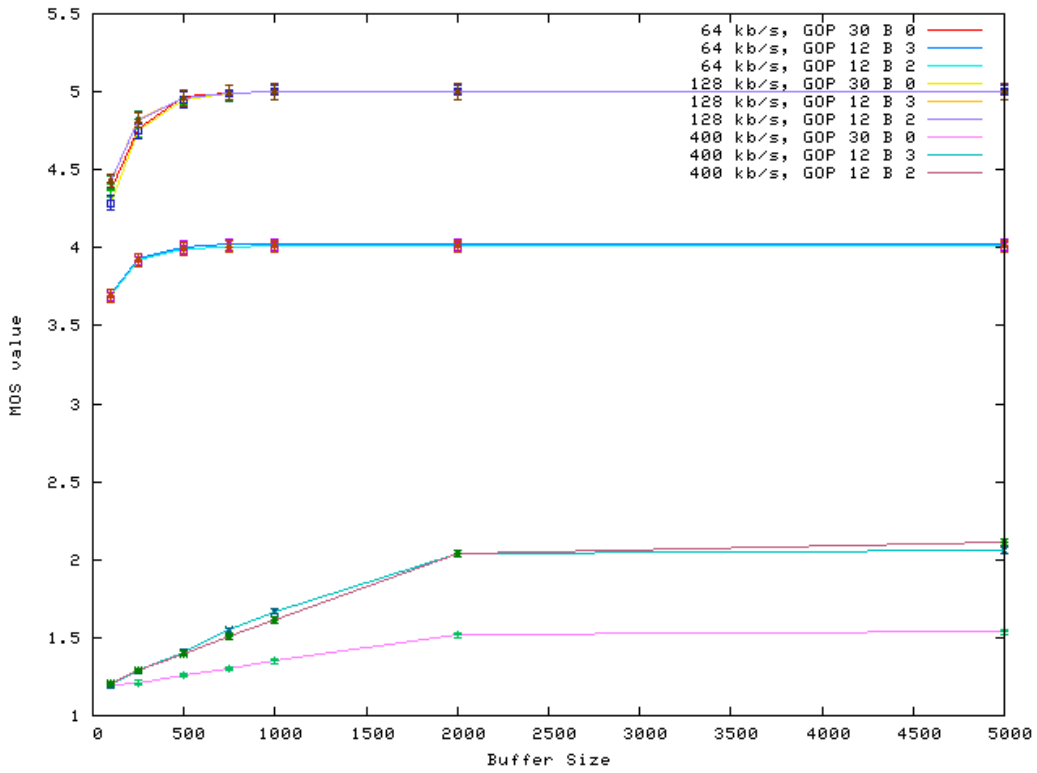


Figure 5.3: $\overline{\text{MOS}}$ of different bitrates and encodings for *akiyo* on evdo-bad network. Encoding are GOP length 30 with no B frames (GOP 30 B 0), GOP length 12 with 3 B frames (GOP 12 B 3) and GOP length 12 with 2 B frames (GOP 12 B 2).

at 400 kb/s, a fraction less than the CDMA2000 network throughput used in this set of results. The lowest of the three is GOP 30 B 0, one of the few times the encoding gave lower quality than the shorter GOP lengths of GOP 12 B 3 and GOP 12 B 2.

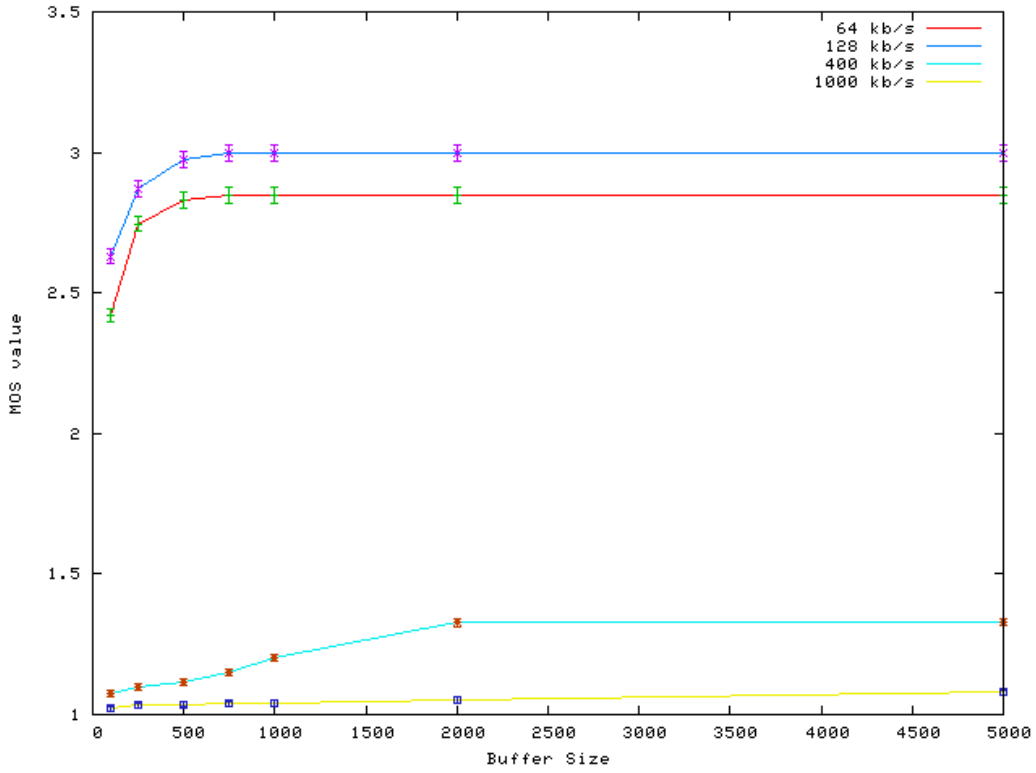


Figure 5.4: $\overline{\text{MOS}}$ of different video bitrates for *coastguard* on evdo-good network with GOP length 30 and 0 B frames encoding.

Figure 5.4 shows different bit rates for the *coastguard* video over the evdo-good network. This shows that the improvement when network capacity is available that the higher bitrate of 128 kb/s has over 64 kb/s. It also show that even when the simulated EVDO link has the highest capacity available, 1.2 Mb/s, that 400 kb/s and 1000 kb/s video bitrates do not improve the quality, and in fact they reduce it substainally.

5.1.2 *Experiment 2*

The results of this experiment show the same general trend up to a limit, over the increasing buffer sizes. The limit appears to be the same for different methods of streaming, as it doesn't appear to change between different transport layers and option, but the limit depends on the video contents and bitrate. The rate that which the quality increases to the limit seems dependent on the network capacity and the transport layer.

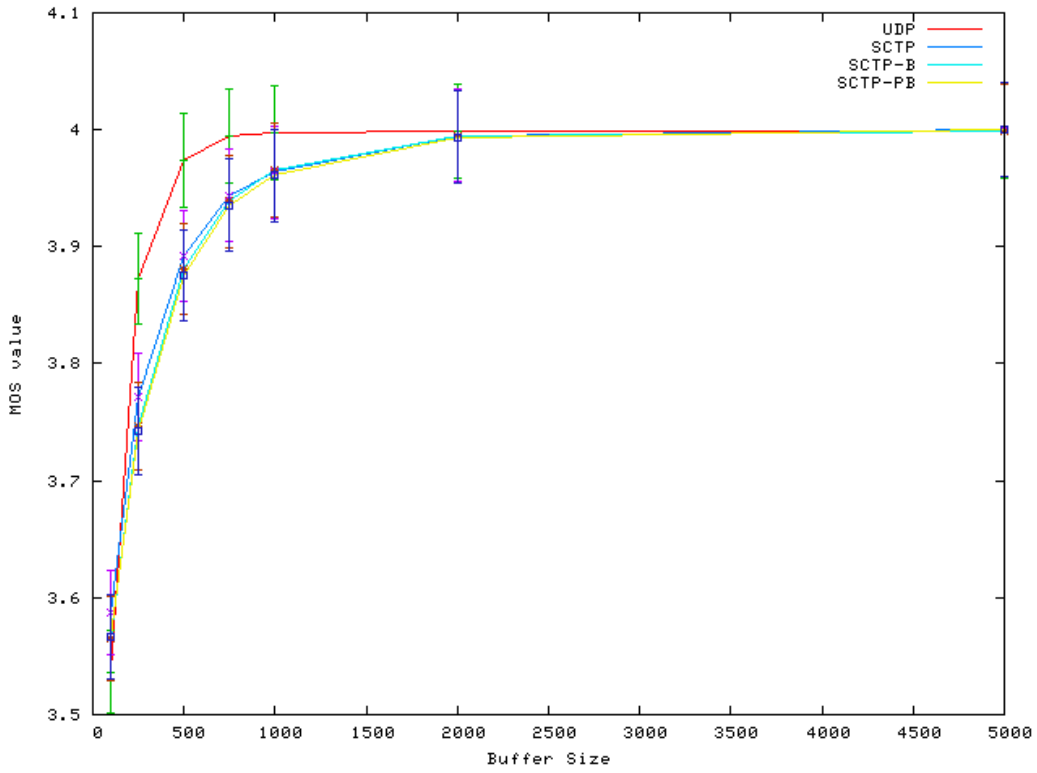


Figure 5.5: $\overline{\text{MOS}}$ of UDP and three types of SCTP streaming of the *news* video at 128 kb/s on evdo-bad network.

A graph typical of streaming using the different protocols is shown in figure 5.5. UDP streaming gives a slightly better quality for buffer sizes between 250 ms and 1000 ms, after which all transport layers and options give the same quality. None of the three options of SCTP are substantially better than any other option.

Figure 5.6 show the quality for the different sizes of video for the *mobile*

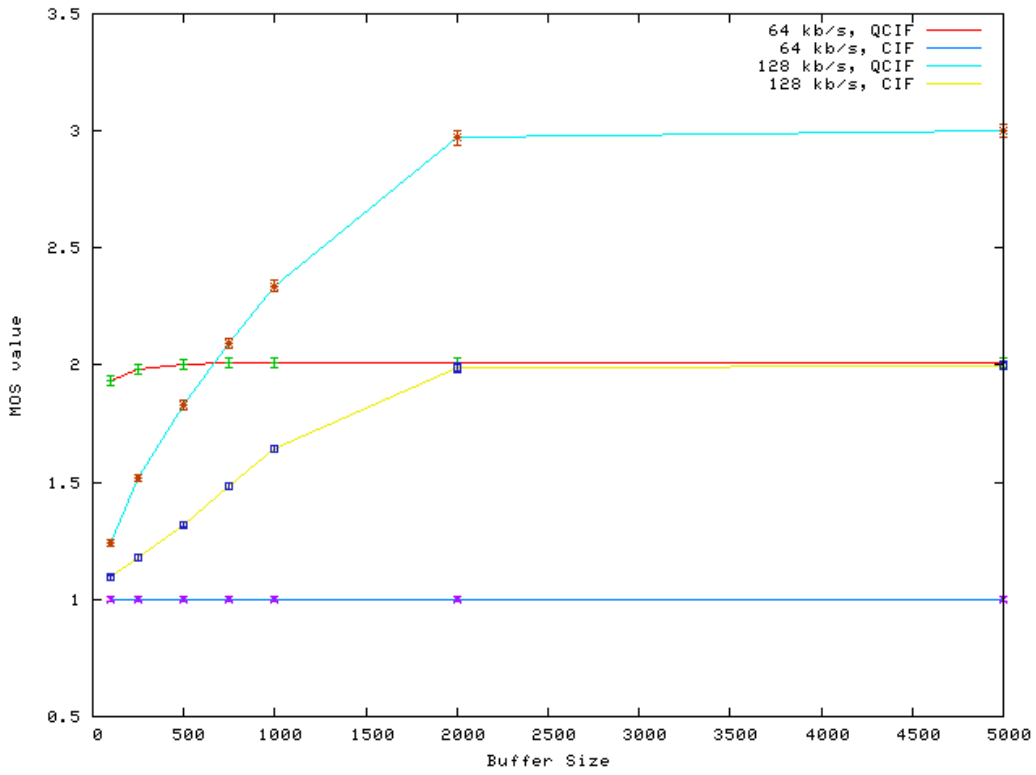


Figure 5.6: $\overline{\text{MOS}}$ of qcif 64 kb/s, cif 64 kb/s, qcif 128 kb/s and cif 128 kb/s using SCTP to stream the *mobile* video over the RTT-good network.

video. At 64 kb/s the CIF size video does not make any improvements over buffer size from the minimum $\overline{\text{MOS}}$ value of 1.0, however at 128 kb/s, the CIF video does improve, be it slower than shown in figure 5.5. QCIF at 64 kb/s makes a slight improvement to about 2.0 MOS score, whereas at 128 kb/s with a buffer size of 100 ms the value is 1.25 increasing to about 3.0 at 2000 ms. The capacity of the CDMA wireless downlink was set to 134 kb/s (repersenting good quality RTT connection), which appears to have a bearing on the rate the quality of the videos at 128 kb/s improves to their limits.

5.1.3 Experiment 3

The results show that the two transport layers are very similar. The results from Experiment 2 to which the results from this experiment are compared to are those for the EVDO-bad network as they should be the closest results. The graphs show error bars for 0.01 for Experiment 2 results, and standard deviation for Experiment 3 results.

Figures 5.7 and 5.8 compare results obtained from measurements in our testbed with the results from Experiment 2.

In figure 5.7, the results of the UDP streaming from Experiment 2 and Experiment 3 are similar, and the margin of error and standard deviation overlap at every buffer size except at 100 ms, where the result of Experiment 3 has a higher $\overline{\text{MOS}}$ value. SCTP in Experiment 3 outpreforms the performance of SCTP from Experiment 2 from 100 ms to about 2000 ms. The testbed results show the quality reaching the limit $\overline{\text{MOS}}$ score of 3.0 with a buffer size of 250 ms, slightly outpreforming UDP streaming in the testbed, but within the standard deviation.

Figure 5.8 shows the advantage the testbed experiment had over the simulation—the testbed EVDO link was running at about 516 kb/s, verses 480 kb/s for the simulation in experiment 2.

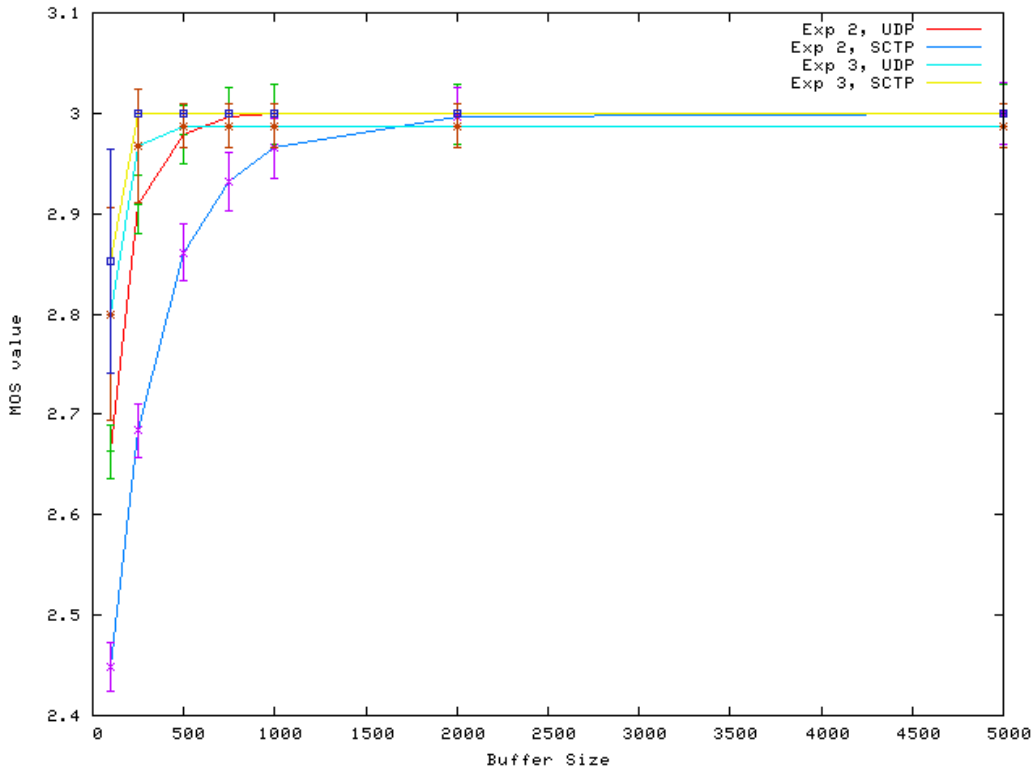


Figure 5.7: $\overline{\text{MOS}}$ of experiment 2 and 3 results for *coastguard* CIF at 128 kb/s.

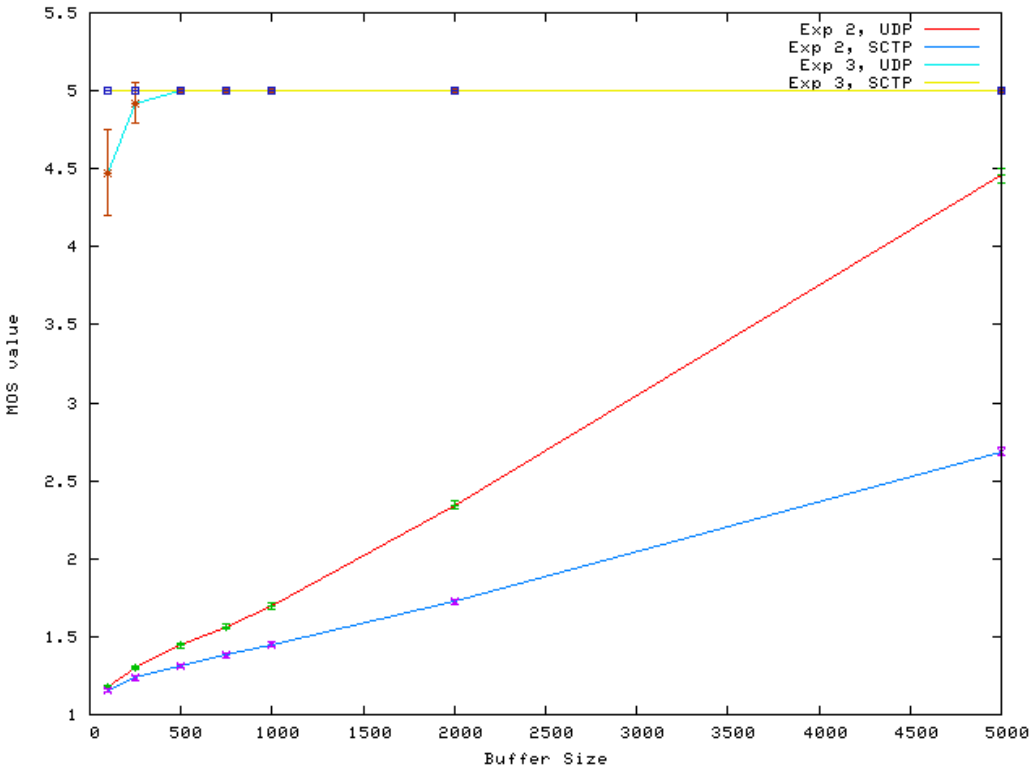


Figure 5.8: $\overline{\text{MOS}}$ of experiment 2 and 3 results for *akiyo* CIF at 400 kb/s.

5.2 Discussion

5.2.1 Experiment 1

As can be seen in figure 5.2, the long GOP appears to have a greater influence on quality. A quick test with IB3PB3 (GOP length 8), confirmed that longer GOP sequences appear to have a higher quality. The transmitted quality is a reflection of the static quality ($\overline{\text{PSNR}}$).

The general trend up to a limit over increasing buffer sizes is caused by variations in the size of frames, and therefore also the bit rate of the video. At small buffer sizes in particular, it may take longer to receive all the packets of the first frame than the size of the buffer allows, causing the first frame to be incomplete, lowering the quality of, and delaying the arrival of, subsequent frames. This can also occur again at later I frames in the video sequence, as they are the largest frame type.

One of the assumptions was found to be violated by the implementation of the topology in NS2. The shared link in the wired network did drop packets when the video bit rate was high, because the data rate for that link was less than the background traffic and the video traffic. This lowered the received quality of the videos at higher bit rates. This is evident in figure 5.4, where the EVDO bitrate is 1248 kb/s and the video bitrate is 400 kb/s—the video should have the highest quality but is the second lowest. This was fixed in Experiment 2.

5.2.2 Experiment 2

It appears that there is no difference in the received video quality between the various options of SCTP. The main mechanism in effect is the flow control, and with no packet loss in the wired network and very low numbers of failures of RLP to recover from wireless losses, very few retransmissions occur that could benefit from partial relability.

UDP provides better quality video than SCTP, especially with lower buffer sizes. The difference in quality of the received videos between SCTP and UDP shows the effect of SCTP's flow control. The flow control in SCTP limits the number of packets sent as to not saturate any of the links in the

network, whereas UDP sends packets regardless. This is particularly everdent in the datarates close to the wireless speed with small buffer sizes, where SCTP detects the increased queueing delay for the CDMA2000 link due to RLP retransmissions and slows the rate packets are sent slightly. Decreasing the rate delays the packets more, and end up missing the buffer, decreasing the quality. At larger buffer sizes, and when the video data rate is less than the CDMA2000 data rate, UDP and SCTP perform similarly.

The smaller frame sizes appear to perform better, but since the QCIF videos are compared to the QCIF original YUV, there is not a fair comparison with CIF videos. A visual comparison performed by several students indicated that for some of the videos—in particular those with lower PSNR—had a better subjective quality for the QCIF-sized video than the CIF size. To make a fair comparison using $\overline{\text{MOS}}$ values between videos of QCIF and CIF size, one must take about 0.4–0.8 off the $\overline{\text{MOS}}$ value for QCIF to make it comparable to the same video at the CIF size.

The quality limit appears to be the same as the full static quality of the video, as it is constant across transport layers. This indicates the lossless transfer of the video in time to not under-flow the buffer.

5.2.3 Experiment 3

The peak data rate was for all runs about 63 kB/s (516 kb/s). Figure 5.9 shows the interface usage graph for the CDMA2000 network interface for a set of experimental runs and two preceding runs. The qcif and cif runs at 128 kb/s the smallest two peaks, and the next run at 400 kb/s nearly reaches the peak rate. The 1000 kb/s rate hits the limit and never exceeds it. For SCTP, the 1000 kb/s run is much longer, and it transfers all the data with reliability, but the other runs are not significantly different between UDP and SCTP.

The different videos have slightly different peaks in data rate, due to the encoding trying to get the best quality out of the average bitrate.

The results are similar for both UDP and SCTP. This could in part be because the CDMA2000 link rate is 516 kb/s, significantly higher than the 480 kb/s used in simulations, and much higher 400 kb/s for the highest video

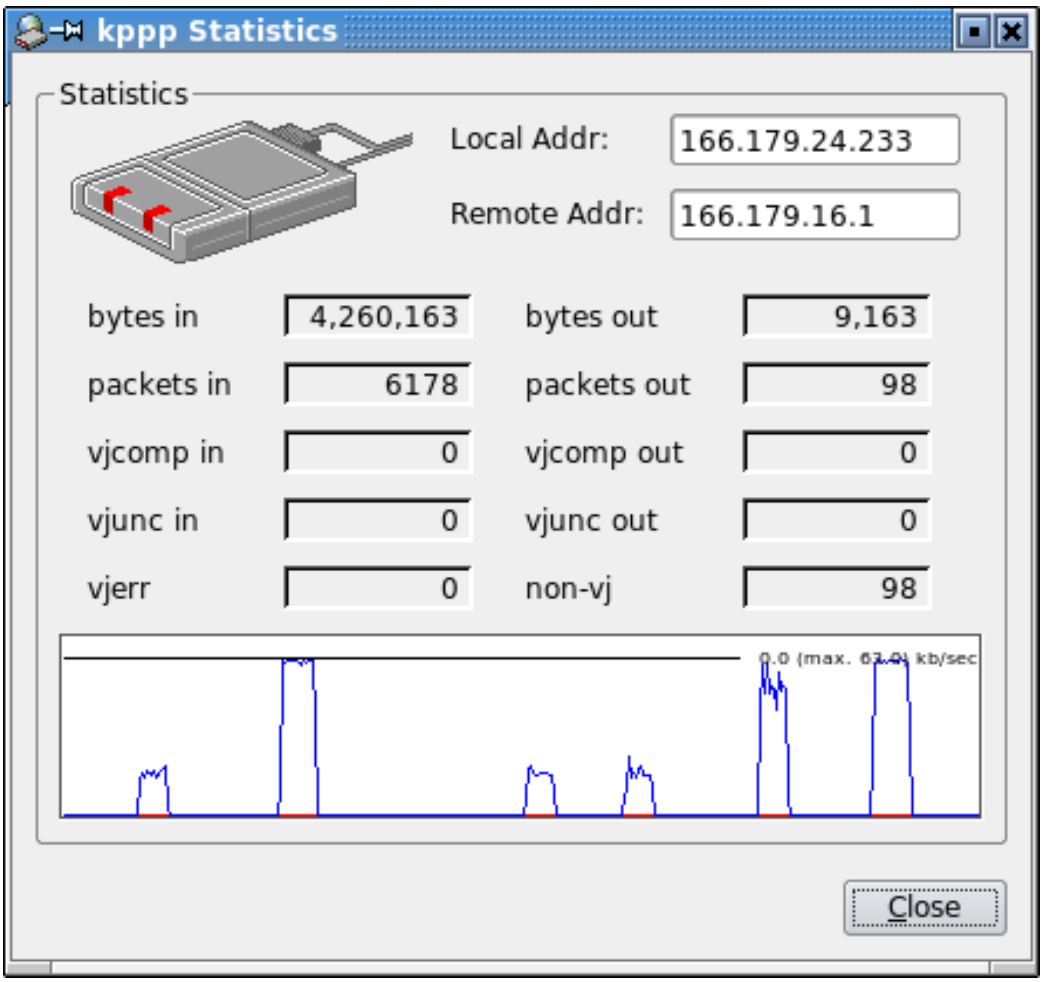


Figure 5.9: Snapshot of the CDMA network interface usage during experiment run, taken on the mobile terminal

under the link rate. Also because of the higher link rate, the results for 400 kb/s are much higher than the simulation gave. For video bit rates of 1000 kb/s, the MOS for both UDP and SCTP is very low, however with a larger buffer size (approximately 12 seconds) SCTP would provide perfect video quality, whereas UDP cannot.

5.2.4 General discussion

Generalised B frames available in H.264/AVC, as mentioned in section 2.1.3, were not used in the simulations because they are not used by default in `x264`. While turning this feature on does improve the quality of the video, it does break the simple dependency model used. While I frames are still independent, any of the previous configurable number of frames (typically 5, 20 or 25) may be depended on by the current P or B frame. This can elevate B and P frames to being more important (i.e., have more dependent frames) than the classical model indicates.

The enhancement layers were not used in this research because the video transmitter lacked a feedback mechanism to decide what to do with the enhancement layers. It is unlikely that with a feedback mechanism the video server would have removed or truncated the enhancement layers (if encoded) except in experiment 3 with video bit rate of 1000 kb/s, because the video bit rate is in excess of the network throughput.

Chapter VI

Conclusions

6.1 Conclusions

The results of Experiment 1 revealed that, while there was no statistically significant difference in quality between the encodings GOP 12 B 3 and GOP 12 B 2 when transmitted over UDP, the GOP 30 B 0 encoding offered better quality. This reflects the quality of the raw encodings as shown by the ($\overline{\text{PSNR}}$).

The results of Experiment 2 revealed that all the SCTP options gave the same quality. UDP streaming was much better at lower buffer sizes, but the difference decreased as the buffer size increased until the difference was statistically insignificant. Flow control in SCTP slowed the departure of packets, requiring larger buffers.

Experiment 3 confirmed that our simulations well represented operations of the real network. Because non-controlable parameters (the wireless link rate) were not the same, the results don't exactly match. The error model for the wireless link used in the simulations might be dropping more packets than the real network does, thus our simulations could be considered as studies of network performance in worse case conditions than in the real world.

6.1.1 Overall Conclusions

The results presented previously in Chapter 5 show, that in networks with very low to no packet loss, there is no advantage to using SCTP instead of UDP, and the congestion control is a limiting factor. The CDMA2000 network using RLP has very close to no packet loss and therefore if the video server is closely connected to the base station, and the links between are not congested, then nearly no packet loss will occur between the video server and

the mobile device.

If the video server is further away, and packets are lost or experience considerable jitter, then a video proxy near the base station could be more useful as it can control the jitter and the proxy would have more bandwidth available to it for retransmissions from the remote video server. The remote video server could use SCTP, or another transport layer, to provide retransmission. The video proxy acts somewhat like a larger buffer to smooth out jitter and provide opportunities for retransmission. When multiple clients request the same video through the proxy, the proxy can reduce the data required from the remote video server, and thereby can increase the quality of service to all clients. This is a form of application level multicast.

6.2 Future Work

There is scope for further work in transport layers for video streaming over wireless networks.

While this thesis did not adapt the rate of the videos as they were streamed, H.264 and other video encoding standards support what is termed "Scalable Video", where the bitrate can be easily reduced from full by not sending parts of the stream. Some research has been carried out into using scalable video; see for example [13]. Possible future work could look at using the scalable and fine-grained scalable profiles of MPEG 4 in H.264 for adapting the bit rate of the video to the current network conditions. This adaption, by matching the video bit rate to the usable data transfer rate should provide higher received quality than guessing the data rate and streaming a video with a lower bit rate. The properties of the measurement process and the changes in bitrate will need to be explored.

Datagram Congestion Control Protocol (DCCP)[11], a relatively new (standardised March 2006) standard transport layer from the IETF, has design features that should make is attractive for video streaming[12]. It has less overhead than SCTP, and should provide good feedback for driving video bitrate adaption for the network congestion and error rates. Application layer or RTCP feedback could be used to provide end to end retransmission, if the receiver has a large enough buffer to handle it.

Due to the increasing use of cellphones, the issues related with the scalability and capacity of any multimedia system, used by mobile users, are becoming very critical elements of any such system's design. The use of storage of videos, the processing power required to send them and the network capacity, all need to be taken into consideration when developing such systems. The processing power needed becomes important in particular when scalable video is used due to the extra video server side processing. It might be possible with clever packet level scheduling to avoid inducing jitter into parallel streams leaving the video server, thereby also reducing the total network capacity needed. Another possible topic for future work would be to work out a formula or rule-of-thumb for how many servers and how much network capacity is needed to serve a given number of clients.

Appendix A

Results

A.1 Experiment 1

This section gives the numerical results for Experiment 1.

The table below shows the $\overline{\text{PSNR}}$ for encoded video, before transmission, for the two videos at the four bitrates for the different encoding settings of GOP length and number of B frames.

Video	bitrate	$\overline{\text{PSNR}}$		
		GOP 30 B 0	GOP 12 B 3	GOP 12 B 2
	kb/s			
akiyo	64	39.287	37.059	37.058
	128	42.236	40.713	40.729
	400	46.284	45.666	45.665
	1000	49.427	49.104	49.061
coastguard	64	27.621	27.396	27.387
	128	29.594	29.432	29.415
	400	33.144	33.080	33.071
	1000	36.879	36.895	36.883

The following four tables (Tables A.1–A.4) show the overall Mean Opinion Score ($\overline{\text{MOS}}$) for the *Akiyo* video for the given network (RTT-bad, RTT-good, EVDO-bad, and EVDO-good) and buffer sizes (100, 250, 500, 750, 1000, 2000, 5000 ms), and each encoding: GOP 30 B 0, GOP 12 B 3, and GOP 12 B 2. The tables are for the video bit rates of 64kb/s, 128kb/s, 400kb/s, and 1000kb/s.

network	Buffer ms	$\overline{\text{MOS}}$ Value GOP 30 B 0	GOP 12 B 3	GOP 12 B 2
RTT-bad	100	4.22545	3.6602	3.67066
	250	4.717	3.89294	3.88787
	500	4.9437	3.99536	3.98662
	750	4.99017	4.016	4.00683
	1000	4.9974	4.01912	4.00918
	2000	4.99828	4.01955	4.00984
	5000	4.99828	4.01955	4.00984
RTT-good	100	4.42454	3.69206	3.67822
	250	4.81305	3.91399	3.90128
	500	4.96515	4.00465	3.99555
	750	4.99292	4.01681	4.00719
	1000	4.9978	4.01901	4.00942
	2000	4.9978	4.01945	4.00942
	5000	4.9978	4.01945	4.00942
EVDO-bad	100	4.3702	3.69833	3.69133
	250	4.75929	3.92635	3.91972
	500	4.96639	4.00639	3.99648
	750	4.99441	4.02	4.00862
	1000	4.99876	4.02	4.01
	2000	4.99876	4.02	4.01
	5000	4.99876	4.02	4.01
EVDO-good	100	4.24241	3.62997	3.61359
	250	4.76367	3.93427	3.92015
	500	4.95754	4.00554	3.99545
	750	4.99321	4.01831	4.00763
	1000	4.99755	4.01997	4.00948
	2000	4.99792	4.01997	4.00985
	5000	4.99792	4.01997	4.00985

Table A.1: Experiment 1 results: *Akiyo*, video bitrate = 64 Kb/s

network	Buffer	$\overline{\text{MOS}}$ Value		
		GOP 30 B 0	GOP 12 B 3	GOP 12 B 2
	ms			
RTT-bad	100	1.33528	1.16194	1.16167
	250	1.38204	1.18189	1.24679
	500	1.39657	1.31457	1.32275
	750	1.39909	1.43897	1.4449
	1000	1.43793	1.48794	1.50003
	2000	1.85847	1.9208	1.91993
	5000	2.99017	3.02371	3.03443
RTT-good	100	1.92408	2.27333	2.37005
	250	2.72964	3.10152	3.15292
	500	4.1149	4.18109	4.18741
	750	4.75838	4.73357	4.73814
	1000	4.94323	4.92766	4.93517
	2000	4.99823	4.99501	4.99895
	5000	4.99831	4.99542	4.99897
EVDO-bad	100	4.28895	4.41727	4.42838
	250	4.74708	4.82336	4.81697
	500	4.95157	4.96023	4.95779
	750	4.99092	4.99241	4.99401
	1000	4.99613	4.99799	4.99844
	2000	4.99729	4.99851	4.99879
	5000	4.99729	4.99851	4.99879
EVDO-good	100	4.22878	4.38271	4.38244
	250	4.73539	4.8325	4.83092
	500	4.94985	4.96163	4.95961
	750	4.98786	4.99403	4.99809
	1000	4.99484	4.99847	4.99809
	2000	4.99484	4.99924	4.9988
	5000	4.99484	4.99924	4.9988

Table A.2: Experiment 1 results: *Akiyo*, video bitrate = 128 Kb/s

network	Buffer ms	$\overline{\text{MOS}}$ Value GOP 30 B 0	GOP 12 B 3	GOP 12 B 2
RTT-bad	100	1.01366	1.0032	1.01739
	250	1.02471	1.00645	1.02666
	500	1.03396	1.0072	1.02829
	750	1.04013	1.0065	1.02833
	1000	1.04325	1.00725	1.02829
	2000	1.04717	1.0097	1.04076
	5000	1.04998	1.0111	1.05056
RTT-good	100	1.12248	1.13206	1.15072
	250	1.1691	1.15538	1.15888
	500	1.18503	1.16053	1.16327
	750	1.19171	1.2156	1.21603
	1000	1.19331	1.24565	1.25173
	2000	1.2031	1.26944	1.28115
	5000	1.25476	1.42286	1.42486
EVDO-bad	100	1.19251	1.1929	1.20667
	250	1.21221	1.28677	1.2892
	500	1.25729	1.40933	1.39657
	750	1.30266	1.54635	1.50689
	1000	1.34973	1.66919	1.61282
	2000	1.519	2.03827	2.03767
	5000	1.53855	2.06482	2.11018
EVDO-good	100	1.19219	1.18677	1.20501
	250	1.21314	1.27931	1.28542
	500	1.25373	1.40193	1.39029
	750	1.30527	1.53751	1.49874
	1000	1.35759	1.66315	1.60908
	2000	1.51456	2.03179	2.03326
	5000	1.53539	2.06055	2.10794

Table A.3: Experiment 1 results: *Akiyo*, video bitrate = 400 Kb/s

network	Buffer	$\overline{\text{MOS}}$ Value		
		GOP 30 B 0	GOP 12 B 3	GOP 12 B 2
	ms			
RTT-bad	100	1.0003	1.0002	1.0002
	250	1.0003	1.0002	1.0002
	500	1.0003	1.0004	1.0005
	750	1.0003	1.0005	1.0006
	1000	1.0003	1.0005	1.0006
	2000	1.0003	1.0005	1.0006
	5000	1.0003	1.0006	1.0008
RTT-good	100	1.0015	1.0013	1.0018
	250	1.003	1.0024	1.0031
	500	1.0032	1.0031	1.0031
	750	1.0032	1.0031	1.0031
	1000	1.0032	1.0031	1.0032
	2000	1.0032	1.0038	1.0039
	5000	1.0034	1.0041	1.005
EVDO-bad	100	1.0326	1.0346	1.0327
	250	1.0353	1.0406	1.0398
	500	1.0361	1.0488	1.0473
	750	1.0388	1.0506	1.0511
	1000	1.0403	1.0541	1.0558
	2000	1.0465	1.0717	1.0771
	5000	1.0715	1.11335	1.1476
EVDO-good	100	1.034	1.03	1.0339
	250	1.0357	1.0351	1.0394
	500	1.0363	1.0389	1.0476
	750	1.0393	1.0414	1.0525
	1000	1.0421	1.0445	1.0559
	2000	1.0489	1.064	1.0761
	5000	1.0763	1.1282	1.1469

Table A.4: Experiment 1 results: *Akiyo*, video bitrate = 1000 Kb/s

The following four tables (Tables A.5–A.8) show the overall Mean Opinion Score ($\overline{\text{MOS}}$) for the *Coastguard* video for the given network (RTT-bad, RTT-good, EVDO-bad, and EVDO-good) and buffer sizes (100, 150, 500, 750, 1000, 2000, 5000 ms), and each encoding: GOP 30 B 0, GOP 12 B 3, and GOP 12 B 2. The tables are for the video bit rates of 64kb/s, 128kb/s, 400kb/s, and 1000kb/s.

network	Buffer	MOS Value		
		GOP 30 B 0	GOP 12 B 3	GOP 12 B 2
	ms			
RTT-bad	100	2.4596	2.17445	2.19179
	250	2.71125	2.59438	2.60424
	500	2.82464	2.79165	2.78889
	750	2.84483	2.8246	2.82395
	1000	2.84932	2.82933	2.82395
	2000	2.84944	2.82978	2.8298
	5000	2.84944	2.82978	2.8298
RTT-good	100	2.53111	2.47046	2.48088
	250	2.74157	2.74594	2.74764
	500	2.83284	2.81336	2.81104
	750	2.84688	2.82813	2.82835
	1000	2.84916	2.82985	2.82974
	2000	2.84916	2.82985	2.82974
	5000	2.84916	2.82985	2.82974
EVDO-bad	100	2.51832	2.49259	2.50647
	250	2.73362	2.75269	2.76561
	500	2.8271	2.81113	2.81522
	750	2.84508	2.82781	2.82799
	1000	2.84898	2.8299	2.8298
	2000	2.84911	2.8299	2.8298
	5000	2.84911	2.8299	2.8298
EVDO-good	100	2.42045	2.43036	2.43615
	250	2.7464	2.75567	2.75835
	500	2.8284	2.81126	2.81352
	750	2.84684	2.82668	2.8276
	1000	2.84903	2.82938	2.82983
	2000	2.84912	2.82988	2.82983
	5000	2.84912	2.82988	2.82983

Table A.5: Experiment 1 results: *Coastguard*, video bitrate = 64 Kb/s

network	Buffer ms	$\overline{\text{MOS}}$ Value GOP 30 B 0	GOP 12 B 3	GOP 12 B 2
RTT-bad	100	1.0444	1.0647	1.0648
	250	1.0744	1.0778	1.0791
	500	1.1337	1.0967	1.0949
	750	1.1829	1.151	1.1522
	1000	1.1991	1.1631	1.1671
	2000	1.4106	1.3668	1.377
	5000	2004	1.9494	1.9695
RTT-good	100	1.46585	1.20046	1.24056
	250	2.08685	1.69977	1.77916
	500	2.70861	2.52675	2.55275
	750	2.92427	2.87141	2.86948
	1000	2.98464	2.97276	2.97073
	2000	2.99969	2.99987	2.99961
	5000	2.99969	2.99987	2.99962
EVDO-bad	100	2.6213	2.67628	2.68007
	250	2.86317	2.91522	2.9121
	500	2.96933	2.97917	2.98
	750	2.99148	2.99465	2.9977
	1000	2.99434	2.99643	2.99974
	2000	2.99434	2.99643	2.99974
	5000	2.99441	2.99643	2.99974
EVDO-good	100	2.6298	2.63813	2.94399
	250	2.86925	2.90953	2.91112
	500	2.9755	2.97877	2.97421
	750	2.9968	2.99753	2.99519
	1000	2.99889	2.99897	2.99638
	2000	2.99889	2.99927	2.99638
	5000	2.99889	2.99927	2.99638

Table A.6: Experiment 1 results: *Coastguard*, video bitrate = 128 Kb/s

network	Buffer	$\overline{\text{MOS}}$ Value		
		GOP 30 B 0	GOP 12 B 3	GOP 12 B 2
	ms			
RTT-bad	100	1.0009	1.0013	1.0014
	250	1.0013	1.0015	1.0015
	500	1.0019	1.0019	1.0021
	750	1.0021	1.0026	1.0023
	1000	1.0029	1.0033	1.0031
	2000	1.004	1.0042	1.0037
	5000	1.005	1.0063	1.0069
RTT-good	100	1.0302	1.0329	1.0328
	250	1.0485	1.0497	1.0495
	500	1.0728	1.0781	1.0787
	750	1.0907	1.0877	1.0877
	1000	1.1033	1.0912	1.0912
	2000	1.1186	1.1353	1.1351
	5000	1.1397	1.1966	1.1979
EVDO-bad	100	1.0793	1.08503	1.08399
	250	1.105	1.14147	1.1423
	500	1.1189	1.19193	1.1955
	750	1.1467	1.23635	1.24292
	1000	1.1963	1.28763	1.29367
	2000	1.3279	1.5234	1.52254
	5000	1.33005	1.60433	1.59628
EVDO-good	100	1.07633	1.08389	1.08494
	250	1.09965	1.14329	1.14282
	500	1.11435	1.19591	1.19421
	750	1.1494	1.24149	1.24143
	1000	1.2014	1.29306	1.29314
	2000	1.3268	1.52103	1.52464
	5000	1.3283	1.59346	1.60561

Table A.7: Experiment 1 results: *Coastguard*, video bitrate = 400 Kb/s

network	Buffer ms	$\overline{\text{MOS}}$ Value GOP 30 B 0	GOP 12 B 3	GOP 12 B 2
RTT-bad	100	1.0002	1.0003	1.0002
	250	1.0002	1.0003	1.0002
	500	1.0003	1.0004	1.0003
	750	1.0003	1.0005	1.0004
	1000	1.0004	1.0005	1.0004
	2000	1.0005	1.0005	1.0004
	5000	1.0005	1.0005	1.0004
RTT-good	100	1.0011	1.0012	1.0016
	250	1.002	1.002	1.0027
	500	1.0028	1.0033	1.0039
	750	1.0036	1.0041	1.0044
	1000	1.0038	1.0041	1.0045
	2000	1.0038	1.0043	1.0045
	5000	1.0044	1.0059	1.007
EVDO-bad	100	1.0216	1.0207	1.0218
	250	1.0342	1.0336	1.0331
	500	1.0371	1.037	1.0357
	750	1.0382	1.0438	1.0403
	1000	1.0416	1.05	1.0471
	2000	1.052	1.0694	1.0666
	5000	1.0791	1.1341	1.1314
EVDO-good	100	1.0205	1.0219	1.0228
	250	1.0328	1.0336	1.0341
	500	1.0367	1.0355	1.0359
	750	1.0376	1.0433	1.0428
	1000	1.042	1.0486	1.0489
	2000	1.052	1.0677	1.0685
	5000	1.0786	1.1299	1.1279

Table A.8: Experiment 1 results: *Coastguard*, video bitrate = 1000 Kb/s

A.2 Experiment 2

This section gives the numerical results for Experiment 2.

The following four tables (Tables A.9–A.12) shows the overall Mean Opinion Score ($\overline{\text{MOS}}$) for the *Akiyo* video for the given size (QCIF and CIF), video bitrate (64, 128, 400 and 1000 Kb/s) and buffer sizes (100, 250, 500, 750, 1000, 2000, and 5000 ms), and each of the transport methods: UDP, SCTP with full reliability, SCTP with partial reliable B frames, and SCTP with partial reliable B and P frames. The tables are for the network types of RTT-bad, RTT-good, EVDO-bad, and EVDO-good.

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	64	100	4.46132	1.80151	1.78373	1.78146
		250	4.81055	2.44832	2.42666	2.41932
		500	4.96947	3.68655	3.66261	3.64533
		750	4.99606	4.5655	4.56035	4.54225
		1000	4.99956	4.90795	4.90325	4.89688
		2000	4.99988	4.9992	4.99964	4.9996
		5000	4.99988	4.9996	4.99996	5
cif	64	100	3.65183	1.41457	1.41008	1.40701
		250	3.88588	1.94819	1.96981	1.93636
		500	3.99853	2.973	2.96981	2.95869
		750	4.01744	3.67088	3.67338	3.66646
		1000	4.01965	3.94182	3.94591	3.94208
		2000	4.01982	4.01865	4.01954	4.01956
		5000	4.01982	4.01904	4.01989	4.01998

Table A.9: Experiment 2 results: *Akiyo*, network type = RTT-bad

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	64	100	4.48875	4.6907	4.70042	4.70305
		250	4.85163	4.87573	4.86532	4.88397
		500	4.97373	4.9688	4.96812	4.97497
		750	4.9958	4.99441	4.99538	4.9989
		1000	4.9999	4.99923	4.99817	4.99953
		2000	4.9999	5	5	5
		5000	4.9999	5	5	5
cif	64	100	3.63572	3.8069	3.78718	3.82293
		250	3.90408	3.92462	3.92126	3.9514
		500	3.99628	4.00231	4.00658	4.0148
		750	4.01599	4.0134	4.01847	4.01995
		1000	4.01947	4.01502	4.01977	4.01995
		2000	4.01947	4.01505	4.02	4.02
		5000	4.01947	4.01505	4.02	4.02
qcif	128	100	1.5562	1.35485	1.35799	1.3551
		250	1.93657	1.50119	1.49889	1.49661
		500	3.13088	1.86245	1.85958	4.8557
		750	4.24166	2.26538	2.2536	2.25141
		1000	4.77205	2.93164	2.90864	2.90401
		2000	4.99827	4.89805	4.88818	4.88188
		5000	4.99866	4.99975	4.99962	4.99967
cif	128	100	2.24252	1.45313	1.4483	1.43849
		250	2.99869	1.80832	1.80384	1.79013
		500	4.09809	2.32835	2.32459	2.29787
		750	4.6859	2.91419	2.91049	2.87653
		1000	4.9188	3.56749	3.56042	3.52728
		2000	4.9994	4.94924	4.94762	4.94373
		5000	4.99945	4.99567	4.99994	4.99991

Table A.10: Experiment 2 results: *Akiyo*, network type = RTT-good

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	128	100	4.41874	4.44432	4.40594	4.41633
		250	4.82976	4.66323	4.64159	4.64803
		500	4.95976	4.84631	4.83526	4.83977
		750	4.99199	4.92194	4.91947	4.92027
		1000	4.99679	4.95586	4.95761	4.95533
		2000	4.99741	4.99006	4.99154	4.99402
		5000	4.99741	4.99623	4.99655	4.99938
cif	128	100	4.41044	4.51499	4.51595	4.5162
		250	4.82816	4.70651	4.71049	4.71468
		500	4.96691	4.86289	4.87413	4.8739
		750	4.99387	4.92027	4.93387	4.93212
		1000	4.99833	4.95055	4.95935	4.96399
		2000	4.9987	4.99403	4.99663	4.99595
		5000	4.9987	4.99998	4.99998	4.99938
cif	400	100	1.17283	1.15535	1.15598	1.15442
		250	1.3015	1.23528	1.23647	1.235
		500	1.44255	1.31233	1.31269	1.31072
		750	1.5638	1.37953	1.3797	1.37792
		1000	1.69673	1.44829	1.44721	1.44602
		2000	2.34333	1.72581	1.72463	1.71966
		5000	4.4585	2.68785	2.69435	2.68567

Table A.11: Experiment 2 results: *Akiyo*, network type = EVDO-bad

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
cif	400	100	1.16863	1.14204	1.13963	1.14029
		250	1.2983	1.21872	1.21629	1.2158
		500	1.44803	1.29436	1.28918	1.28924
		750	1.56643	1.3598	1.35505	1.35494
		1000	1.6961	1.41872	1.41519	1.41538
		2000	2.33807	1.67044	1.66513	1.6619
		5000	4.44177	2.51396	2.51398	2.50447
cif	1000	100	1.0671	1.0321	1.03197	1.0298
		250	1.1379	1.08807	1.08643	1.0838
		500	1.16397	1.13103	1.1297	1.13093
		750	1.1765	1.15405	1.1556	1.15535
		1000	1.22427	1.17373	1.17297	1.173
		2000	1.34127	1.24823	1.2503	1.2503
		5000	1.4362	1.4805	1.48343	1.4843

Table A.12: Experiment 2 results: *Akiyo*, network type = EVDO-good

The following four tables (Tables A.13–A.16) shows the overall Mean Opinion Score ($\overline{\text{MOS}}$) for the *Coastguard* video for the given size (QCIF and CIF), video bitrate (64, 128, 400 and 1000 Kb/s) and buffer sizes (100, 250, 500, 750, 1000, 2000, and 5000 ms), and each of the transport methods: UDP, SCTP with full reliability, SCTP with partial reliable B frames, and SCTP with partial reliable B and P frames. The tables are for the network types of RTT-bad, RTT-good, EVDO-bad, and EVDO-good.

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	64	100	2.44569	1.10661	1.10622	1.10528
		250	2.88478	1.29979	1.29328	1.29633
		500	3.08311	1.69719	1.66738	1.68425
		750	3.12224	2.33111	2.24617	2.31355
		1000	3.12868	2.85609	2.99462	2.84074
		2000	3.12961	3.12877	3.12869	3.12934
		5000	3.12961	3.12927	3.12975	3.12997
cif	64	100	2.12318	1.01858	1.01857	1.01877
		250	2.56697	1.10563	1.10467	1.1036
		500	2.78261	1.46965	1.45471	1.44668
		750	2.82173	2.36998	2.33882	2.32417
		1000	2.82837	2.73034	2.71324	2.70705
		2000	2.82972	2.82945	2.82991	2.8298
		5000	2.82972	2.82966	2.83	2.82999

Table A.13: Experiment 2 results: *Coastguard*, network type = RTT-bad

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	64	100	2.69572	2.39009	2.39957	2.40727
		250	3.0345	2.86856	2.87941	2.8819
		500	3.105	3.06882	3.07374	3.07154
		750	3.12713	3.1184	3.11982	3.11925
		1000	3.12925	3.12737	3.12786	3.12778
		2000	3.1297	3.1297	3.12994	3.12994
		5000	3.1297	3.1297	3.12995	3.12995
cif	64	100	2.43243	2.12377	2.14759	2.14407
		250	2.74831	2.59512	2.60735	2.60833
		500	2.80756	2.77768	2.78065	2.78215
		750	2.82774	2.81924	2.82049	2.82084
		1000	2.82965	2.8275	2.82818	2.82804
		2000	2.82987	2.82947	2.82999	2.83
		5000	2.82987	2.82947	2.83	2.83
qcif	128	100	1.40503	1.17797	1.16972	1.17152
		250	1.6643	1.34416	1.33312	1.33667
		500	2.46276	1.56754	1.55749	1.56327
		750	3.27105	1.8467	1.83269	1.83931
		1000	3.71677	2.2683	2.24535	2.25467
		2000	3.949	3.79489	3.78301	3.78907
		5000	3.94944	3.94995	3.94997	3.94991
cif	128	100	1.18721	1.07299	1.07246	1.07219
		250	1.60002	1.13385	1.13113	1.13009
		500	2.39318	1.36947	1.36529	1.36195
		750	2.80672	1.68202	1.67774	1.67542
		1000	2.9519	2.03045	2.02367	2.02408
		2000	2.99981	2.95322	2.96009	2.95722
		5000	2.99983	2.99994	2.99999	2.99994

Table A.14: Experiment 2 results: *Coastguard*, network type = RTT-good

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	128	100	3.42798	3.22948	3.21655	3.21572
		250	3.82845	3.5247	3.51527	3.5091
		500	3.9164	3.75839	3.74357	3.73528
		750	3.94792	3.85605	3.84352	3.83788
		1000	3.94963	3.90013	3.89372	3.88837
		2000	3.94963	3.94443	3.94301	3.94377
		5000	3.94963	3.94988	3.95	3.94992
cif	128	100	2.66338	2.44861	2.44389	2.45295
		250	2.90997	2.68383	2.6763	2.68215
		500	2.9787	2.86132	2.86054	2.85955
		750	2.99597	2.93131	2.93323	2.93088
		1000	2.99963	2.96508	2.96456	2.96324
		2000	2.99963	2.99632	2.99609	2.99607
		5000	2.99963	2.99979	2.9998	2.9998
cif	400	100	1.0772	1.04808	1.04757	1.04718
		250	1.13835	1.07408	1.07424	1.0741
		500	1.2231	1.10562	1.10355	1.10344
		750	1.3018	1.14554	1.14344	1.14249
		1000	1.3751	1.18494	1.18143	1.17979
		2000	1.71685	1.34017	1.33287	1.33129
		5000	3.12975	1.93797	1.91018	1.90098

Table A.15: Experiment 2 results: *Coastguard*, network type = EVDO-bad

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
cif	400	100	1.07543	1.04703	1.04688	1.04705
		250	1.1378	1.07247	1.07211	1.07228
		500	1.22007	1.09817	1.09801	1.09759
		740	1.30535	1.13367	1.1322	1.13358
		1000	1.37473	1.16717	1.16606	1.16556
		2000	1.72163	1.307	1.30297	1.30284
		5000	3.13687	1.81632	1.80664	1.79593
cif	1000	100	1.021	1.0136	1.0143	1.014
		250	1.0359	1.0255	1.02565	1.02545
		500	1.0681	1.04325	1.0443	1.04485
		750	1.0952	1.0616	1.0629	1.0622
		1000	1.1145	1.07805	1.07625	1.0785
		2000	1.226	1.1322	1.1259	1.13125
		5000	1.3443	1.30505	1.29545	1.30025

Table A.16: Experiment 2 results: *Coastguard*, network type = EVDO-good

The following four tables (Tables A.17–A.20) shows the overall Mean Opinion Score ($\overline{\text{MOS}}$) for the *Mobile* video for the given size (QCIF and CIF), video bitrate (64, 128, 400 and 1000 Kb/s) and buffer sizes (100, 250, 500, 750, 1000, 2000, and 5000 ms), and each of the transport methods: UDP, SCTP with full reliability, SCTP with partial reliable B frames, and SCTP with partial reliable B and P frames. The tables are for the network types of RTT-bad, RTT-good, EVDO-bad, and EVDO-good.

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	64	100	1.84822	1.33211	1.32966	1.324
		250	1.94904	1.4869	1.48415	1.47988
		500	1.99954	1.71851	1.70595	1.71818
		750	2.0083	1.89105	1.88416	1.89006
		1000	2.0098	1.97764	1.97439	1.97504
		2000	2.00994	2.00904	2.00993	2.00988
		5000	2.00994	2.00911	2.01	2.00993
cif	64	100	1	1	1	1
		250	1	1	1	1
		500	1	1	1	1
		750	1	1	1	1
		1000	1	1	1	1
		2000	1	1	1	1
		5000	1	1	1	1

Table A.17: Experiment 2 results: *Mobile*, network type = RTT-bad

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	64	100	1.84865	1.9345	1.9392	1.9436
		250	1.9616	1.9818	1.9837	1.9872
		500	2.0013	2.0028	2.0067	2.0064
		750	2.0084	2.0084	2.01	2.01
		1000	2.01	2.01	2.01	2.01
		2000	2.01	2.01	2.01	2.01
		5000	2.01	2.01	2.01	2.01
cif	64	100	1	1	1	1
		250	1	1	1	1
		500	1	1	1	1
		750	1	1	1	1
		1000	1	1	1	1
		2000	1	1	1	1
		5000	1	1	1	1
qcif	128	100	1.69377	1.24289	1.25057	1.248
		250	2.04354	1.52003	1.21853	1.51888
		500	2.4889	1.83037	1.8292	1.82529
		750	2.78788	2.09555	2.09496	2.09229
		1000	2.9342	2.33792	2.33828	2.34003
		2000	2.99944	2.97141	2.97141	2.97421
		5000	2.99952	3	3	3
cif	128	100	1.29797	1.09383	1.09432	1.0931
		250	1.49019	1.17941	1.17863	1.18086
		500	1.75584	1.32565	1.32084	1.32634
		750	1.90684	1.48553	1.48219	1.48967
		1000	1.97326	1.64629	1.64255	1.64984
		2000	1.99979	1.98677	1.9877	1.98317
		5000	1.99983	1.99982	1.99998	2

Table A.18: Experiment 2 results: *Mobile*, network type = RTT-good

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	128	100	2.67092	2.72496	2.72417	2.73709
		250	2.9105	2.84262	2.84272	2.85305
		500	2.9812	2.9263	2.92829	2.93247
		750	2.99633	2.95938	2.96288	2.96419
		1000	2.99903	2.97586	2.97677	2.97746
		2000	2.99903	2.99719	2.99568	2.99725
		5000	2.99903	2.99998	2.99997	3
cif	128	100	1.8267	1.87716	1.88256	1.8807
		250	1.9557	1.9288	1.93158	1.93128
		500	1.9883	1.97085	1.97037	1.9695
		750	1.9984	1.98327	1.98592	1.98305
		1000	2	1.9891	1.9918	1.98892
		2000	2	1.99685	1.99885	1.99865
		5000	2	1.9987	2	2
cif	400	100	1.0804	1.06655	1.06567	1.06657
		250	1.13065	1.08203	1.08292	1.08286
		500	1.21375	1.12793	1.13004	1.12834
		750	1.3095	1.17496	1.17363	1.17724
		1000	1.39585	1.22319	1.22231	1.2231
		2000	1.6556	1.41117	1.41585	1.40678
		5000	2.3982	1.88204	1.88962	1.87869

Table A.19: Experiment 2 results: *Mobile*, network type = EVDO-bad

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
cif	400	100	1.0807	1.06259	1.06304	1.06289
		250	1.12865	1.08073	1.08095	1.08005
		500	1.21105	1.12009	1.11768	1.11689
		750	1.2984	1.15841	1.15529	1.15797
		1000	1.38935	1.20097	1.19705	1.19793
		2000	1.65335	1.37105	1.36609	1.36911
		5000	2.39005	1.80057	1.80269	1.79457
cif	1000	100	1.04134	1.0215	1.0204	1.0218
		250	1.07099	1.0401	1.0386	1.04137
		500	1.09527	1.06605	1.0643	1.06687
		750	1.10059	1.0822	1.07925	1.0826
		1000	1.10856	1.09355	1.09082	1.09247
		2000	1.18398	1.13375	1.13305	1.13767
		5000	1.33065	1.31555	1.30583	1.32177

Table A.20: Experiment 2 results: *Mobile*, network type = RTT-bad

The following four tables (Tables A.21–A.24) shows the overall Mean Opinion Score ($\overline{\text{MOS}}$) for the *News* video for the given size (QCIF and CIF), video bitrate (64, 128, 400 and 1000 Kb/s) and buffer sizes (100, 250, 500, 750, 1000, 2000, and 5000 ms), and each of the transport methods: UDP, SCTP with full reliability, SCTP with partial reliable B frames, and SCTP with partial reliable B and P frames. The tables are for the network types of RTT-bad, RTT-good, EVDO-bad, and EVDO-good.

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	64	100	3.56932	1.9344	1.95938	1.95533
		250	3.84215	2.33248	2.3646	2.35978
		500	3.9752	3.103	3.15142	3.15799
		750	3.99664	3.69621	3.73286	3.74901
		1000	3.99941	3.92703	3.94424	3.94517
		2000	3.99962	3.9998	3.99927	3.99984
		5000	3.99962	4	3.99952	4
cif	64	100	2.7554	1.35435	1.35752	1.3467
		250	2.91115	1.71674	1.71322	1.71004
		500	2.98704	2.316	2.3112	2.30494
		750	2.99842	2.74806	2.75713	2.74657
		1000	2.99915	2.93508	2.94193	2.93651
		2000	2.99921	2.99953	2.99917	2.99948
		5000	2.99921	3	2.99977	2.99998

Table A.21: Experiment 2 results: *News*, network type = RTT-bad

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	64	100	3.59626	3.80983	3.79401	3.8134
		250	3.8808	3.94185	3.91844	3.9328
		500	3.97738	3.99585	3.98179	3.9813
		750	3.99558	4	3.99864	3.99305
		1000	3.99973	4	4	3.9999
		2000	3.99973	4	4	4
		5000	3.99973	4	4	4
cif	64	100	2.74123	2.8288	2.82752	2.81285
		250	2.92285	2.93254	2.94391	2.91753
		500	2.9876	2.99045	2.98862	2.98032
		750	2.9965	2.99871	2.99846	2.99244
		1000	2.99915	2.9996	2.9996	2.99388
		2000	2.99915	2.99995	3	2.99533
		5000	2.99915	3	3	2.99536
qcif	128	100	2.29878	1.52342	1.52845	1.5217
		250	3.02363	1.96822	1.98093	1.96971
		500	4.11371	2.50766	2.52539	2.51664
		750	4.69171	3.05543	3.07499	3.05696
		1000	4.90889	3.69716	3.73288	3.72279
		2000	4.98896	4.96125	4.96543	4.96277
		5000	4.98895	4.98981	4.98904	4.98981
cif	128	100	2.00772	1.27693	1.27524	1.28328
		250	2.6209	1.62525	1.63568	1.65395
		500	3.44871	2.04553	2.06055	2.07062
		750	3.82687	2.4865	2.51031	2.52276
		1000	3.95777	2.96835	3.00694	3.01385
		2000	3.99831	3.9706	3.98175	3.97857
		5000	3.99835	3.99996	3.99994	3.9996

Table A.22: Experiment 2 results: *News*, network type = RTT-good

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
qcif	128	100	4.4556	4.49842	4.51112	4.50221
		250	4.84002	4.69229	4.70525	4.70276
		500	4.96318	4.8613	4.86558	4.86269
		750	4.98415	4.92957	4.9289	4.92547
		1000	4.98714	4.9548	4.95563	4.95381
		2000	4.98776	4.98676	4.9851	4.98739
		5000	4.98776	4.98972	4.98976	4.98987
cif	128	100	3.53657	3.58775	3.56488	3.56673
		250	3.87287	3.77138	3.74618	3.7421
		500	3.97403	3.89193	3.8813	3.87565
		750	3.99489	3.94348	3.93879	3.93561
		1000	3.99736	3.96365	3.96557	3.96119
		2000	3.99855	3.99515	3.99421	3.99397
		5000	3.99855	3.99963	3.99942	3.99998
cif	400	100	1.15652	1.12721	1.12712	1.12835
		250	1.27713	1.16217	1.16046	1.16244
		500	1.45149	1.2567	1.25993	1.26045
		750	1.66926	1.34839	1.35564	1.35274
		1000	1.91062	1.43734	1.44529	1.44389
		2000	2.54257	1.79622	1.81295	1.80796
		5000	4.54817	2.77855	2.80125	2.79243

Table A.23: Experiment 2 results: *News*, network type = EVDO-bad

Size	Bitrate	Buffer Size	$\overline{\text{MOS}}$ value over transport			
			UDP	SCTP	SCTP-B	SCTP-PB
cif	400	100	1.16131	1.12233	1.12419	1.1215
		250	1.28519	1.15621	1.15822	1.15605
		500	1.4519	1.23157	1.2373	1.23103
		750	1.66102	1.31557	1.32471	1.31407
		1000	1.89802	1.39088	1.40389	1.38845
		2000	2.5371	1.71393	1.72708	1.71298
		5000	4.54689	2.59021	2.62841	2.60608
cif	1000	100	1.01944	1.01017	1.0102	1.01013
		250	1.0702	1.0301	1.0311	1.0315
		500	1.13629	1.079	1.07935	1.07943
		750	1.15144	1.1084	1.11199	1.1095
		1000	1.16427	1.13153	1.13553	1.13657
		2000	1.3001	1.22123	1.21165	1.2149
		5000	1.43669	1.4858	1.47633	1.477

Table A.24: Experiment 2 results: *News*, network type = EVDO-good

A.3 Experiment 3

This section gives the numerical results for Experiment 3.

The following four tables (Tables A.25–A.28) shows the overall Mean Opinion Score ($\overline{\text{MOS}}$) for each of the videos (*Akiyo*, *Coastguard*, *News*, *Mobile*) for the given size (QCIF and CIF), video bitrate (64, 128, 400 and 1000 Kb/s) and the transport methods: UDP, SCTP with full reliability. There is a table for each video sources.

Size	Bitrate	Buffer Size	UDP		SCTP	
			$\overline{\text{MOS}}$	S.D.	$\overline{\text{MOS}}$	S.D.
qcif	128	100	4.6075	0.082272	4.8675	0.105445
		250	4.9925	0.01299	5	0
		500	5	0	5	0
		750	5	0	5	0
		1000	5	0	5	0
		2000	5	0	5	0
		5000	5	0	5	0
cif	128	100	4.6475	0.15287	4.9775	0.038971
		250	4.9175	0.082576	5	0
		500	5	0	5	0
		750	5	0	5	0
		1000	5	0	5	0
		2000	5	0	5	0
		5000	5	0	5	0
cif	400	100	4.4725	0.2749	5	0
		250	4.9225	0.13423	5	0
		500	5	0	5	0
		750	5	0	5	0
		1000	5	0	5	0
		2000	5	0	5	0
		5000	5	0	5	0
cif	1000	100	1.17	0	1.17	0
		250	1.17	0	1.17	0
		500	1.3125	0.01479	1.3125	0.042647
		750	1.35	0	1.35	0
		1000	1.41	0.01225	1.395	0.020616
		2000	1.7	0	1.7	0
		5000	2.5325	0.12597	2.405	0.065383

Table A.25: Experiment 3 testbed results: *Akiyo*

Size	Bitrate	Buffer Size	UDP		SCTP	
			$\overline{\text{MOS}}$	S.D.	$\overline{\text{MOS}}$	S.D.
qcif	128	100	3.6125	0.17079	3.775	0.113689
		250	3.95	0	3.95	0
		500	3.95	0	3.95	0
		750	3.95	0	3.95	0
		1000	3.95	0	3.95	0
		2000	3.95	0	3.95	0
		5000	3.95	0	3.95	0
cif	128	100	2.8	0.10630	2.8525	0.111215
		250	2.9675	0.056291	3.0	0
		500	2.9875	0.021650	3.0	0
		750	2.9875	0.021650	3.0	0
		1000	2.9875	0.021650	3.0	0
		2000	2.9875	0.021650	3.0	0
		5000	2.9875	0.021650	3.0	0
cif	400	100	3.0875	0.28856	3.2875	0.355673
		250	3.5675	0.16021	3.66	0
		500	3.64	0.03464	3.66	0
		750	3.64	0.03464	3.66	0
		1000	3.64	0.03464	3.66	0
		2000	3.64	0.03464	3.66	0
		5000	3.64	0.03464	3.66	0
cif	1000	100	1.0225	0.00433	1.0325	0.012990
		250	1.06	0.007071	1.065	0.015
		500	1.12	0	1.1175	0.004330
		750	1.15	0.007071	1.1575	0.021650
		1000	1.23	0.007071	1.23	0.017321
		2000	1.38	0.007071	1.3775	0.004330
		5000	2.0425	0.058041	1.9675	0.021651

Table A.26: Experiment 3 testbed results: *Coastguard*

Size	Bitrate	Buffer Size	UDP		SCTP	
			$\overline{\text{MOS}}$	S.D.	$\overline{\text{MOS}}$	S.D.
qcif	128	100	4.79	0.11080	4.8875	0.1188223
		250	4.99	0	4.99	0
		500	4.99	0	4.99	0
		750	4.99	0	4.99	0
		1000	4.99	0	4.99	0
		2000	4.99	0	4.99	0
		5000	4.99	0	4.99	0
cif	128	100	3.7425	0.11648	3.865	0.135185
		250	4.0	0	3.96	0.069282
		500	4.0	0	3.99	0.017351
		750	4.0	0	3.99	0.017351
		1000	4.0	0	3.99	0.017351
		2000	4.0	0	3.99	0.017351
		5000	4.0	0	3.99	0.017351
cif	400	100	4.0325	0.982557	4.96	0.069282
		250	4.3875	1.060881	5	0
		500	4.3875	1.060881	5	0
		750	4.3875	1.060881	5	0
		1000	4.4275	0.991599	5	0
		2000	4.9825	0.030311	5	0
		5000	4.982	0.030311	5	0
cif	1000	100	1.055	0.025981	1.09	0
		250	1.16	0	1.16	0
		500	1.16	0	1.16	0
		750	1.32	0	1.32	0
		1000	1.3625	0.025860	1.33	0
		2000	1.64	0	1.6375	0.0043301
		5000	2.545	0.095525	2.3975	0.073612

Table A.27: Experiment 3 testbed results: *News*

Size	Bitrate	Buffer Size	UDP		SCTP	
			$\overline{\text{MOS}}$	S.D.	$\overline{\text{MOS}}$	S.D.
qcif	128	100	2.7825	0.059739	2.945	0.05895
		250	3.0	0	3.0	0
		500	3.0	0	3.0	0
		750	3.0	0	3.0	0
		1000	3.0	0	3.0	0
		2000	3.0	0	3.0	0
		5000	3.0	0	3.0	0
cif	128	100	1.9125	0.042057	1.9625	0.037666
		250	2.0	0	2.0	0
		500	2.0	0	2.0	0
		750	2.0	0	2.0	0
		1000	2.0	0	2.0	0
		2000	2.0	0	2.0	0
		5000	2.0	0	2.0	0
cif	400	100	2.755	0.110567	2.4625	0.846592
		250	3.0	0	2.4625	0.846592
		500	3.0	0	2.505	0.857365
		750	3.0	0	2.52	0.831384
		1000	3.0	0	2.52	0.831384
		2000	3.0	0	2.88	0.207846
		5000	3.0	0	2.98	0.034641
cif	1000	100	1.0325	0.0043301	1.1	0
		250	1.0975	0.0043301	1.1	0
		500	1.1	0	1.1	0
		750	1.2025	0.012990	1.21	0
		1000	1.21	0	1.21	0
		2000	1.44	0	1.44	0
		5000	2.000	0.083367	1.97	0

Table A.28: Experiment 3 testbed results: *Mobile*

Appendix B

Source Code

This appendix lays out the source code for programs written or modified for use in this research.

B.1 `udp2tcpdump` *and* `sctp2tcpdump`

Two programs were written in Perl to transform the `ns2` output trace file into the two tcpdump files needed for `etmp4`. Two separate programs were needed because of of the connection setup in SCTP that is not present in UDP.

This is the source code for udp2tcpdump.pl.

```
#! /usr/bin/perl

($txfilename, $rxfilename, $pktheader) = @ARGV;

open(TXFILE,"> $txfilename");
open(RXFILE,"> $rxfilename");

#for each line (event) in the trace file
while(<STDIN>){
  ($type, $time, $src, $dest, $proto, $size, $dummy1, $flowid,
      $dummy3, $dummy4, $dummy5, $pktid) = split(/ /, $_);
  chomp($pktid);

  #find UDP flows with id 0

  if($flowid eq "0" && $proto eq "udp"){
```

```
    if($type eq "+" && $src eq "0" && $dest eq "1"){

        #queued to leave the source, add to TX tcpdump file
        print "sent $time $pktid\n";
        printf TXFILE "%-16f id %-16d udp %-16d\n",
                     ($time, $pktid, $size);

    }else{
      if($type eq "r" && $dest eq "7" && $src eq "6"){

          #received at sink,, add to RX tcpdump file
          print "recv $time $pktid\n";
          printf RXFILE "%-16f id %-16d udp %-16d\n",
                       ($time, $pktid, $size);

      }
    }
  }
}

close(TXFILE);
close(RXFILE);
```

This is the source code for sctp2tcpdump.pl.

```
#! /usr/bin/perl

($txfilename, $rxfilename, $pktheader, $first, $framefile,
                        $mtu, $hint) = @ARGV;

open(TXFILE,"> $txfilename");
open(RXFILE,"> $rxfilename");

open(FFILE, "$framefile");
```

```
$starting = ($first == 1);

$fpkt = 0;
$fsize = $hint;

%pktid2 = ();
%fpktsize = ();
$fpktnum = 1;
$ftime = 0;

#for each line (event) in the trace file
while(<STDIN>){
  ($type, $time, $src, $dest, $proto, $size, $dummy1, $dummy2,
          $dummy3, $dummy4, $dummy5, $pktid) = split(/ /, $_);
  chomp($pktid);

  # for SCTP events
  if($proto eq "sctp"){
    #queued to leave the source
    if($type eq "+" && $src eq "0" && $dest eq "1"){
      #if part of the 4 way handshake, do nothing
      if($starting && ($size == 56 || $size == 36)){

      }else{
        #remove the size of the packet header (inc sctp chunk header)
        $size = $size - $pktheader;

        #while there are more data chunks in this packet
        while($size > 0){

          #if the current frame has been sent, then get the
          # next frame's size
          if($fsize == 0){
            $frameline = <FFILE>;
```

```
            if(! defined ($frameline)){
              last;
            }
            chomp($frameline);
            ($dummy1, $dummy2, $fsize, $dummy3, $dummy4) =
                                  split("\t", $frameline);
            #if the hint has just been sent, then remove it's size
            # from the first normal frame
            if($hint != 0){
              $fsize = $fsize - $hint;
              $hint = 0;
            }
#           print "fsize = $fsize\n";

            #advance the ideal frame transmission time
            $ftime = $ftime + 1/25;
          }
          print "Sending size $size, fsize $fsize\n";

          #if the frame or the remaining part of the
          # frame fits in the packet, then write it
          # was sent in the TX tcpdump file
          if($fsize <= $mtu && $size >= $fsize){
            if($ftime == 0){
              $ftime = $time;
            }
#print "sent $time $pktid\n";
            printf TXFILE "%-16f id %-16d udp %-16d\n",
                           ($ftime, $fpktnum, $fsize);

            #trace frame/packet size and id for receiving side
            $fpktsize{$fpktnum} = $fsize;
            if(exists $pktid2{$pktid}){
              $pktid2{$pktid}[1] = $fpktnum;
```

```
            }else{
              $pktid2{$pktid} = [ ($fpktnum, $fpktnum) ];
            }
            $size = $size - $fsize;
            $fsize = 0;
            $fpktnum++;

          #if frame is larger than packet, send a packet
          # full, and reduce the amount of frame to send
          # Written into TX tcpdump file.
          }elsif($fsize >= $mtu && $size >= $mtu){
            if($ftime == 0){
              $ftime = $time;
            }
#print "sent $time $pktid\n";
            printf TXFILE "%-16f id %-16d udp %-16d\n",
                              ($ftime, $fpktnum, $mtu);

            #same tracking as above
            $fpktsize{$fpktnum} = $mtu;
            if(exists $pktid2{$pktid}){
              $pktid2{$pktid}[1] = $fpktnum;
            }else{
              $pktid2{$pktid} = [ ($fpktnum, $fpktnum) ];
            }
            $size = $size - $mtu;
            $fsize = $fsize - $mtu;
            $fpktnum++;
          }else{
            print "remaining size $size, fsize $fsize\n";
            $size = 0;
          }
        }
      }
```

```
    }else{

      #if received at the sink
      if($type eq "r" && $dest eq "7" && $src eq "6"){
        #if 4 way handshake, do nothing
        if($starting && ($size == 56 || $size == 36)){
          $starting = !($size == 36);
        }else{
#print "recv $time $pktid\n";

          #locate id from packet id for each frame (part) in the packet
          if(exists $pktid2{$pktid}){
            foreach $i ($pktid2{$pktid}[0]..$pktid2{$pktid}[1]){
              #write received into RX tcpdump file using size of that
              # frame part
              printf RXFILE "%-16f id %-16d udp %-16d\n",
                            ($time, $i, $fpktsize{$i});
            }
          }
        }
      }
    }
  }
}

close(TXFILE);
close(RXFILE);
close(FFILE);
```

B.2 sctp2udpdump

The captured tcpdump files in Experiment 3 when using SCTP could not be read by `etmp3`. To transform them, and to remove the connection setup, the program `sctp2udpdump` was written in Perl.

This is the source code for sctp2udpdump.pl.

```
#! /usr/bin/perl

($txfilename, $rxfilename, $txin, $rxin, $mtu, $hint,
                            $framefile) = @ARGV;

open(TXFILE,"> $txfilename");

open(FFILE, "$framefile");
open(TXIN, "$txin");

$starting = 0;

$pktheader = 48;

$fpkt = 0;
$fsize = $hint;
$savehint = $hint;

%pktid2 = ();
%fpktsize = ();
$fpktnum = 1;
$ftime = 0;

#for each tcpdump record (one on each line) for the
# source node
while(<TXIN>){
  ($time, $dummy6, $dummy7, $dummy8, $dummy9, $dummy10,
          $dummy11, $dummy15, $pktid, $dummy1, $dummy2,
          $dummy3, $dummy4, $dummy5, $dummy12, $dummy13,
          $size, $src, $dummy14, $dest, $proto) =
                                          split(/ /, $_);

  #clean up inputs
```

```
chomp($proto);
$pktid=~s/,//;
$size=~s/\)//;
$src=~s/([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+).*/$1/;
$dest=~s/([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+).*/$1/;

#remove headers from size to make etmp4 happy
$size = ($size - $pktheader);

#if packet is from the video server
if($src == "132.181.9.13"){
  #skip over the 4 way handshake (two packets
  # each way)
  if($starting < 2){
    $starting++;
  }else{
    #while there is still some of the packet
    while($size > 0){
      #Get frame's size if don't have it already
      if($fsize <= 0 || $fsize < 10){
        $frameline = <FFILE>;
        if(! defined ($frameline)){
          last;
        }
        chomp($frameline);
        ($dummy1, $dummy2, $fsize, $dummy3, $dummy4) =
                                split("\t", $frameline);

        #if the hint has just been sent, then remove it's size
        # from the first normal frame
        if($hint != 0){
          $fsize = $fsize - $hint;
          $hint = 0;
        }
```

```
  #advance the ideal frame transmission time
  $ftime = $ftime + 1/25;
}

  #if the frame or the remaining part of the
  # frame fits in the packet, then write it
  # was sent in the TX tcpdump file
  if($fsize <= $mtu && $size >= $fsize){
    if($ftime == 0){
      $ftime = $time;
    }
    printf TXFILE "%-16f id %-16d udp %-16d\n",
                  ($ftime, $fpktnum, $fsize);

    #trace frame/packet size and id for receiving side
    $fpktsize{$fpktnum} = $fsize;
    if(exists $pktid2{$pktid}){
      $pktid2{$pktid}[1] = $fpktnum;
    }else{
      $pktid2{$pktid} = [ ($fpktnum, $fpktnum) ];
    }
    $size = $size - $fsize;
    $fsize = 0;
    $fpktnum++;
  }elsif($fsize >= $mtu && $size >= $mtu){
    #if frame is larger than packet, send a packet
    # full, and reduce the amount of frame to send
    # Written into TX tcpdump file, pretending to be UDP
    if($ftime == 0){
      $ftime = $time;
    }
    printf TXFILE "%-16f id %-16d udp %-16d\n",
                  ($ftime, $fpktnum, $mtu);
```

```
            #trace frame/packet size and id for receiving side
            $fpktsize{$fpktnum} = $mtu;
            if(exists $pktid2{$pktid}){
              $pktid2{$pktid}[1] = $fpktnum;
            }else{
              $pktid2{$pktid} = [ ($fpktnum, $fpktnum) ];
            }
            $size = $size - $mtu;
            $fsize = $fsize - $mtu;
            $fpktnum++;
          }else{
            print "remaining size $size, fsize $fsize\n";
            $size = 0;
          }
        }
      }
  }
}

close(TXFILE);
close(FFILE);
close(TXIN);

open(RXIN, "$rxin");
open(RXFILE,"> $rxfilename");

$starting = 0;

#for each tcpdump record (one on each line) for the
# sink node
while(<RXIN>){
  ($time, $dummy6, $dummy7, $dummy8, $dummy9, $dummy10,
          $dummy11, $dummy15, $pktid, $dummy1, $dummy2,
```

```
          $dummy3, $dummy4, $dummy5, $dummy12, $dummy13,
          $dummy15, $size, $src, $dummy14, $dest, $proto) =
                                              split(/ /, $_);

  #clean up input
  chomp($proto);
  $pktid=~s/,//;
  $size=~s/\)//;
  $src=~s/([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+).*/$1/;
  $dest=~s/([0-9]+\.[0-9]+\.[0-9]+\.[0-9]+).*/$1/;
  $size = ($size - $pktheader);

  if($src == "132.181.9.13"){

    #skip over 4 way handshake
    if($starting < 2){
      $starting++;

    }else{
      if($size > 0){
        #look up packet and write correct information to
        # tcpdump file, pretending to be UDP
        if(exists $pktid2{$pktid}){
          foreach $i ($pktid2{$pktid}[0]..$pktid2{$pktid}[1]){
            printf RXFILE "%-16f id %-16d udp %-16d\n",
                          ($time, $i, $fpktsize{$i});
          }
        }
      }
    }
  }
}

close(RXIN);
```

```
close(RXFILE);
```

B.3 Modifications to `mp4trace` *for SCTP streaming*

Because of difficulties creating an SCTP connections from the video server to the mobile terminal, the socket creation code in socket.c was modified to create a listen socket, block on accepting a new connection until the mobile terminal connected and then stream the video. This worked well and does not affect the result.

The host lookup functions were removed as they were not needed, as the video server waiting for a client to connect before replying with the video data, instead of being told the hostname to stream the video to on the command line.

This is the changes in unified diff format.

```
--- evalvid-2.1/socket.c        2006-02-08 03:30:47.000000000 +1300
+++ evalvid-sctp/socket.c       2006-12-04 14:18:36.000000000 +1300
@@ -9,6 +9,7 @@
   #include <sys/types.h>
   #include <sys/socket.h>
   #include <netinet/in.h>
+  #include <netinet/sctp.h>
   #include <netdb.h>
 #endif

@@ -20,7 +21,6 @@
 /* private */

 static sock_t sock_;
-static struct hostent *host_;
 static struct sockaddr_in addr_;

 /* public */
@@ -35,22 +35,26 @@
       goto SI;
```

```
 #endif

-  if ((sock_ = socket(AF_INET, SOCK_DGRAM, 0)) < 0) goto SI;
+  sock_t listen_;
+
+  if ((listen_ = socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP)) < 0) goto SI;
+  int nodelay = 1;
+  if(setsockopt(listen_, 132 , SCTP_NODELAY, &nodelay, 4) != 0 ) goto SI;

   addr_.sin_family = AF_INET;
   addr_.sin_addr.s_addr = INADDR_ANY;
   addr_.sin_port = htons(p);

-  if ((host_ = gethostbyname(h)) == 0) goto UH;
-
-  memcpy(&addr_.sin_addr, host_->h_addr, host_->h_length);
-
-  if (connect(sock_, (struct sockaddr *) &addr_, sizeof addr_) < 0) goto CF;
+  if (bind(listen_, (struct sockaddr *) &addr_, sizeof addr_) < 0) goto CF;
+
+  if (listen(listen_, 1) < 0) goto CF;
+
+  sock_ = accept(listen_, NULL, NULL);
+  close(listen_);

   return 1;

 SI: seterror(err_SI); goto X;
-UH: seterror(err_UH); goto X;
 CF: seterror(err_CF);
 X:  return 0;
 }
```

Here is the complete modified socket.c file for streaming using SCTP.

```
#include <stdio.h>
#include <string.h>

#if defined(_WIN32)
  #include <winsock2.h>
#elif defined(__linux__) || defined(__APPLE__)
  #include <unistd.h>
  #include <sys/types.h>
  #include <sys/socket.h>
  #include <netinet/in.h>
  #include <netinet/sctp.h>
  #include <netdb.h>
#endif

#include "error.h"
#include "rtp.h"
#include "socket.h"
#include "timing.h"

/* private */

static sock_t sock_;
static struct sockaddr_in addr_;

/* public */

int setdest(char *h, unsigned short p)
{
#if defined(_WIN32)
  WSADATA wsa = {0};

  if (WSAStartup(0x202, &wsa) == 0)
    if (LOBYTE(wsa.wVersion) != 2 || HIBYTE(wsa.wVersion) != 2)
      goto SI;
```

```
#endif

  sock_t listen_;

  if ((listen_ = socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP)) < 0) goto SI;
  int nodelay = 1;
  if(setsockopt(listen_, 132 , SCTP_NODELAY, &nodelay, 4) != 0 ) goto SI;

  addr_.sin_family = AF_INET;
  addr_.sin_addr.s_addr = INADDR_ANY;
  addr_.sin_port = htons(p);

  if (bind(listen_, (struct sockaddr *) &addr_, sizeof addr_) < 0) goto CF;

  if (listen(listen_, 1) < 0) goto CF;

  sock_ = accept(listen_, NULL, NULL);
  close(listen_);

  return 1;

SI: seterror(err_SI); goto X;
CF: seterror(err_CF);
X:  return 0;
}

int sendbuf(unsigned char *p, unsigned l)
{
  int sent;

  while (0 > (sent = send(sock_, (const char *)p, l, 0))) {
    perror("send");
    SLEEP(0);
  }
```

```
  return sent;
}

void cleanup()
{
#if defined(_WIN32)
  WSACleanup();
#endif
}
```

References

[1] A. Argyriou and V. Madisentti. Streaming H.264/AVC video over the Internet. In *First IEEE Consumer Communications and Networking Conference*, pages 169–174. IEEE, January 2004.

[2] A. Balk, M. Sigler, M. Gerla, and M. Sansadidi. Investigation of MPEG4 video streaming over SCTP. In *6th World Multiconference on Systemics, cybernetics, and Informatics*, July 2002.

[3] A. Caro, P. Amer, P. Conrad, and G Heinz. Improving multimedia performance over lossy networks via SCTP. In *Fifth Advanced Telecommunications and Information Distribution Research Program*, March 2001.

[4] P. Conrad, A. Caro, and P. Amer. ReMDoR: Remote multimedia document retrieval over partial order transport. *ACM Transactions on Multimedia*, September 2001.

[5] G. Ewing, K. Pawlikowski, and D. McNickle. Akaroa-2: Exploiting network computing by distributing stochastic simulation. *Proceedings of the 1999 European Simulation Multiconference*, pages 175–181, 1999.

[6] Shaojian Fu, Mohammed Atiquzzaman, and William Ivancic. Evaluation of SCTP for space networks. *IEEE Wireless Communications*, 12(5):54–62, October 2005.

[7] E. Gilbert. Capacity of burst-noise channel. *Bell System Technology Journal*, 39:1253–1266, September 1960.

[8] Sangeun Han. Overview of H.264/MPEG-4 part 10: Advanced video coding (AVC) standard. Lecture notes on website, November 2003. `https://www.ece.ucdavis.edu/~mihaela/lecture11.pdf`.

[9] University of Southern California Information Sciences Institute. RFC 793: Transmission control procotol. In *IETF Internet Standards*. IETF, September 1981.

[10] J. Klaue, B. Rathke, and A. Wolisz. EvalVid - a framework for video transmission and quality evaluation. In *The 13th International Conference on Modelling Techniques and Tools for Computer Preformance Evaluation*, pages 255–272, September 2003.

[11] E. Kohler, M. Handley, and S. Floyd. RFC 4340: Datagram congestion control procotol. In *IETF Internet Standards*. IETF, 2006.

[12] Eddie Kohler, Mark Handley, and Sally Floyd. Designing DCCP: Congestion control without reliability. In *ACM SIGCOMM 2006*, September 2006.

[13] M. Lee, R. Qiao, and K. Bengston. Error-resilient scalable video over the internet. In *Australian Telecommunications Networks and Applications Conference 2006*, December 2006.

[14] Ze-Nian Li and Mark Drew. *Fundamentals of Multimedia*. Pearson Education, 2004.

[15] H. Lin and S. Das. Performance study of link layer and MAC layer protocols to support TCP in 3G CDMA systems. *IEEE Transactions on Mobile Computing*, 4(5):489–501, Sept 2005.

[16] Mei Hwan Loke, Ee Ping Ong, Weisi Lin, Zhongkang Lu, and Susu Yao. Comparison of video quality metrics on multimedia videos. In *2006 IEEE International Conference on Image Processing*, pages 457–460, October 2006.

[17] J. McDougall and S. Miller. Sensitivity of wireless network simulation to a two-state markov model channel approximation. In *2003 IEEE GLOBECOM*, pages 697–701, October 2003.

[18] J. McDougall, Y. Yu, and S. Miller. A statistical approach to developing channel models for network simulations. In *2004 IEEE Wireless Communication and Networking Conference*, pages 1660–1665, March 2004.

[19] M. Molteni and M. Villari. Using SCTP with partial reliability for MPEG-4 multimedia streaming. In *Proc. of BSDCon Europe 2002*, October 2002.

[20] Moving Picture Experts Group, International Organistaion for Standardisation. `http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm`. Website, November 2005.

[21] J. Postel. RFC 768: User datagram procotol. In *IETF Internet Standards*. IETF, August 1980.

[22] Protocol Engineering Laboratory. `http://pel.cis.udel.edu/`. Website, April 2005.

[23] O. Rose. Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems. In *Proceedings of the 20th Annual Conference on Local Computer Networks*, volume 20, pages 397–406, 1995.

[24] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad. RFC 3758: Stream control transmission procotol (SCTP) partial reliability extension. In *IETF Internet Standards*. IETF, 2004.

[25] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Tayor, I. Rytina, M. Kalla, L.Zhang, and V. Paxson. RFC 2960: Stream control transmission procotol. In *IETF Internet Standards*. IETF, 2000.

[26] Telecommunication Networks Group, Faculty of Electrical Engineering and Computer Science, Berlin University of Technology. `http://www.tkn.tu-berlin.de/research/evalvid/`. Website, March 2006.

[27] V. Tralli and M. Zorzi. Markov models for the physical layer block error process in a WCDMA cellular system. *IEEE Transactions on Vechicular Technology*, 54(6):2102–2113, Novemeber 2005.

[28] W. Xiaoyi. General analytical model of RLP in cdma2000 system. In *2005 IEE International Conference on Mobile Technology, Applications and Systems*, November 2005.

[29] M. Zorzi and R. Rao. On the statistics of block errors in bursty channel. *IEEE Transactions on Communications*, 45(6):660–667, June 1997.